

UAH RESEARCH REPORT NO. 407

OMV--A SIMPLIFIED MATHEMATICAL MODEL OF THE
ORBITAL MANEUVERING VEHICLE

NASA-CR-171123
19840021833

Kenneth E. Johnson
Environmental and Energy
Center



The University
Of Alabama
In Huntsville

LIBRARY COPY

JAN 24 1985

LANGLEY RESEARCH CENTER
LIBRARY, NASA
HAMPTON, VIRGINIA

August 1984



NF02664

OMV -- A Simplified Mathematical Model
Of The
Orbital Maneuvering Vehicle
Interim Report

Prepared for
George C. Marshall Space Flight Center
Marshall Space Flight Center
Huntsville, AL 35812

by
Dr. William Teoh
Kenneth Johnson Energy & Environment Center
University of Alabama in Huntsville
Huntsville, AL 35899

Date prepared : August 9, 1984

Contract # NAS8-35670

Abstract

A model of the Orbital Maneuvering Vehicle is presented. In this model, several simplifications have been made. A set of hand controller signals may be used to control the motion of the OMV.

Model verification is carried out using a sequence of tests. The dynamic variables generated by the model is compared, whenever possible, with the corresponding analytical ones. The result of the tests show conclusively that the present model is behaving correctly. Further, this model interfaces properly with the State Vector Transformation module (SVX) developed previously. Correct command sequences are generated by the OMV and SVX system, and these command sequences can be used to drive the flat floor simulation system here at MSFC.

N84-29902[#]

INTRODUCTION

This report discusses the design and implementation of OMV -- a mathematical model of the Orbital Maneuvering Vehicle. To avoid confusion, the term OMV shall be used to mean the mathematical model as well as the software that performs the modelling, while the full term "Orbital Maneuvering Vehicle" shall be used to mean the actual flight hardware.

The Orbital Maneuvering Vehicle can be maneuvered by remote operator control. Its motion is completely specified by its equations of motion. The solution of the equations of motion yields its position $[X,Y,Z]^T$, velocity $[\dot{X},\dot{Y},\dot{Z}]^T$, orientation $[r,p,y]^T$ and their rates $[\dot{r},\dot{p},\dot{y}]^T$ where r , p and y stand for roll, pitch and yaw respectively. From these dynamic quantities, a 14-component state vector can be generated. This state vector contains all the necessary information to completely specify the state of the vehicle in space at any time.

The OMV simulates the motion of the Orbital Maneuvering Vehicle in space. OMV is a software subsystem that is an integral part of the software system used to drive the MSFC flat floor simulation system. In this installation, a set of hand controllers is used to maneuver the OMV (Mathematical model) and the state vector obtained is used as input to a second software module called SVX (the State Vector Transformation module) which transforms it to a suitable set of commands to be transmitted to, and thereby controlling the mobile base on the flat floor. The over-all relation is as shown in Figure 1. As can be seen in this

figure, the OMV module encompasses the vehicle response module as well as the orbital mechanics module. In order to optimize execution speed, these two modules are not implemented as separate entities.

The State Vector Transformation Module has been discussed elsewhere (see Reference 1) and will not be elaborated here. Throughout this report, it is important to bear in mind that the OMV simulates the motion of the Orbital Maneuvering vehicle but otherwise has no physical relationship with the Orbital Maneuvering Vehicle. The mobile base on the flat floor will attempt to move in such a manner that a mock-up module mounted on it will replicate the motion of the Orbital Maneuvering Vehicle, using a set of commands derived from the state vectors generated by OMV. Otherwise the mobile base is not related to the OMV. The mock-up module is not the Orbital Maneuvering Vehicle. One of the objectives of the flat floor system is to simulate docking of the OMV with a target vehicle.

THE OMV MODEL

This report describes a simplified mathematical model of the Orbital Maneuvering Vehicle. A more detailed model is being developed elsewhere at MSFC. In the present model, several simplifications and assumptions have been made. The objective is to develop quickly (and hence the simplification) a model that can be used to drive the flat floor system.

Before discussing the model in any detail, it is necessary

to define the various coordinate systems used in this work.

A. The Local Vertical Frame (LVF)

Imagine a space craft in an orbit around the earth. It is immaterial whether this is the Orbital Maneuvering Vehicle or the target vehicle. LVF is a coordinate system with its origin at the center of mass of this space craft such that Z-axis lies in the plane of the orbit and is directed away from the center of the earth. The Y-axis is chosen to be parallel to the orbital angular momentum vector and X-axis is tangential to the orbit as shown in Figure 2. The position, velocity as well as orientation of the second vehicle are described in LVF and is therefore relative to the orbiting vehicle. Throughout this work, we shall assume that the target vehicle is the orbiting vehicle.

B. OMV Body Frame

This is a body fixed reference frame with its origin fixed at the center of mass of the OMV, and its axes will be denoted by 1, 2 and 3 respectively. Initially, at the start of the simulation, 1, 2 and 3 axes line up with X, Y and Z axes respectively. As can be seen from Figure 3, the axis of symmetry is the 1-axis.

In order to construct the model of the Orbital Maneuvering Vehicle, the following assumptions are made :

1. The OMV is assumed to be a circular disk of constant mass and having a uniform mass distribution. This assumption may seem unreasonable at first glance, but one quickly realizes

that the detail shape of the OMV is unimportant as long as one knows the mass and propulsion characteristics of the Orbital Maneuvering Vehicle. In the present model, the mass characteristics are summarized in Table 1. These figures are taken from the MSFC Preliminary Definition Studies (see Reference 2).

2. The OMV is manipulated using signals from a set of hand controllers. These signal can be classified into two groups. The first group is used to simulate a force acting through the center of mass of the OMV. In other words, one can, from this group of signals, generate an acceleration vector $\mathbf{a} = [a_1, a_2, a_3]^T$ in the body frame. The other group of signals simulates rotations about 1, 2 and 3 axes, namely, a vector $\mathbf{w} = [w_1, w_2, w_3]^T$. Assumptions 1 and 2 mean that detailed knowledge of the shape, thrust level and placement of the thruster and so forth are not really needed. The present control mode is the only mode implemented.
3. Circular orbits are assumed. The altitude of the orbit can be anything from 150 to 1500 nautical miles which is the designed operating range of the Orbital Maneuvering Vehicle.
4. Orbital mechanics is an important part in describing the motion of the OMV and is therefore implemented. Other secondary perturbation effects are totally ignored.
5. The state of the OMV is computed and updated 10 times per second. The period of 0.1 second will be referred to as a major cycle throughout this report.

The equations of motion of the OMV can be discussed in terms of the rotational part and translational part.

Rotational Equations of Motion

The rotational equation of motion can be written as :

$$\tau = \dot{L}$$

where $L = I\omega$ is the angular momentum vector and τ is the applied torque. I is the moment of inertia tensor and ω is the body rate. The solution can be drastically simplified by choosing the body axes 1, 2 and 3 such that I is diagonal (Please see References 3 and 4), that is :

$$I = \begin{bmatrix} I_{11} & 0 & 0 \\ 0 & I_{22} & 0 \\ 0 & 0 & I_{33} \end{bmatrix}$$

Remember that $\omega = [\omega_1, \omega_2, \omega_3]^T$ is obtained from the hand controller signals. The solution of the rotational equations of motion yields ϕ , θ and ψ the three Euler angles. The order and sense of rotation is chosen in the conventional manner (Please see Reference 5), that is :

$$[\phi]_1 [\theta]_3 [\psi]_2$$

To reduce computational overhead, quaternions are used to specify the attitude of the OMV rather than the Euler angles themselves. It has been proven that the two representations are exactly equivalent (Reference 6). A quaternion q may be written as :

$$q = iq_1 + jq_2 + kq_3 + q_4 = [q_1, q_2, q_3, q_4]^T$$

and satisfies the relation

$$q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$$

An object whose attitude is described by the three Euler angles relative to some reference frame can be treated as a single rotation by α about an Euler axis $E = [E_1, E_2, E_3]^T$. Theory has shown that this is the shortest angular path (Reference 7) in the sense that α is less than the algebraic sum of ϕ , θ and ψ . The angle α and the Euler axis can be expressed in terms of the quaternion q as :

$$\cos \frac{\alpha}{2} = q_4$$

$$E = (iq_1 + jq_2 + kq_3) / (q_1^2 + q_2^2 + q_3^2)^{1/2}$$

Since the attitude control system of the OMV can control the roll, pitch and yaw axis independently, we expect the roll, pitch and yaw $[r, p, y]^T$ to be proportional to the respective components of E (Reference 7). In fact, the following relation holds :

$$[r, p, y]^T = [\alpha E_x, \alpha E_y, \alpha E_z]^T$$

Quaternion algebra leads to further computational economy when successive rotations need to be calculated. Let say, at any instant, the attitude of the OMV is specified by the quaternion q_1 relative to some non-rotating frame. Suppose further that an instant later, the vehicle's attitude has changed, having rotated by ϕ , θ and ψ . These angular displacements are measured relative to the rotated body frame. If the new attitude is described

by a second quaternion q_2 , the attitude of the vehicle, relative to the non-rotating frame (References 8,9) is then given by

$$q = q_1 q_2$$

This is an important advantage because if at the beginning of the simulation, the body frame is aligned with the LVF (as specified by the quaternion $q_0 = [0,0,0,1]^T$), then the attitude of the OMV relative to the LVF, after n successive rotations is simply:

$$q = q_0 q_1 q_2 \dots q_n$$

Of course, the attitude of the vehicle after the $n+1$ -th rotation is $q = q_n q_{n+1}$. Thus, the attitude of the vehicle can be computed from the previous quaternions. This recursive property gives rise to quite a computational advantage, especially since there are only four elements in a given quaternion versus the nine elements of a direction cosine matrix.

TRANSLATIONAL EQUATION OF MOTION

The translational equations of motion has been derived in detail in Appendix I, and will not be repeated here. In essence, we seek solutions to a set of three simultaneous, coupled second order differential equations of the form :

$$\begin{aligned}\ddot{X} &= A_x - 2\omega \dot{Z} \\ \ddot{Y} &= A_y - \omega^2 Y \\ \ddot{Z} &= A_z + 2\omega \dot{X} + 3\omega^2 Z\end{aligned}$$

Here, the position and velocity vectors $[X,Y,Z]^T$ and $[\dot{X},\dot{Y},\dot{Z}]^T$

refer to the position and velocity of the OMV relative to the target vehicle, as expressed in Local Vertical Frame. ω is the orbital velocity, and $A = [A_x, A_y, A_z]^T$ is the linear acceleration vector in LVF. Remember that the hand controller signals give rise to an acceleration vector $a = [a_1, a_2, a_3]^T$ in OMV body frame. Thus, one can obtain A from a using the transformation :

$$A = C^{-1}a$$

where C^{-1} is the inverse of the direction cosine matrix which can be derived from the quaternion $q = [q_1, q_2, q_3, q_4]^T$ as:

$$C^{-1} = \begin{bmatrix} q_4 + q_1 - q_2 - q_3 & 2(q_1q_2 - q_3q_4) & 2(q_1q_3 + q_2q_4) \\ 2(q_1q_2 + q_3q_4) & q_4 - q_1 + q_2 - q_3 & 2(q_2q_3 - q_1q_4) \\ 2(q_1q_3 - q_2q_4) & 2(q_2q_3 + q_1q_4) & q_4 - q_1 - q_2 + q_3 \end{bmatrix}$$

It is obviously impractical to seek an analytical solution to the translational equations of motion. Numerical methods must be used. In the present work, the Adam-Bashforth method is used. For this purpose, each major cycle is subdivided into N (normally 10, but see later section) sub-intervals, each of which will be referred to as a minor cycle. It is necessary that the acceleration vector A be computed for each minor cycle, and stored in an acceleration matrix. At the end of N minor cycles, this acceleration matrix is used to obtain the numerical solution for the entire major cycle. A 14-component state vector is then assembled, and their components are listed below :

- S(1) - S(3) -- relative position vector in LVF
- S(4) - S(6) -- relative velocity vector in LVF
- S(7) - S(9) -- angular momentum vector in LVF

S(10) - S(13) -- attitude quaternion

S(14) -- mass in kilograms

The angular momentum vector in LVF can be deduced as follows. Since the body rate $\mathbf{w} = [w_1, w_2, w_3]^T$ is known, one can calculate L_B in body frame using the relation (Reference 10):

$$L_B = I\mathbf{w}$$

$$L = C^{-1}L_B$$

where C^{-1} is the inverse of the direction cosine matrix.

The state vector serves as input to the State Vector Transformation module (SVX). This module has been designed and implemented (Reference 1) and will not be repeated here. For completeness, a copy of the updated report is included in Appendix 2.

System Design and Implementation

The design and implementation of the present system is best discussed in the following sub-sections :

A) Hand Controllers

The hand controllers allow the operator to manipulate the Orbital Maneuvering Vehicle in terms of translation and attitude. In the present system, hand controller signals are used to maneuver the OMV. The hardware is configured to provide 12 bits of information. The first 6 bits pertain to translation, while the remaining 6 bits pertain to attitude control. During development, the 12 bits are simulated by reading them from a disk file (HNDSGL.DAT) as 12 single digit integers. This process is carried

out in a subprogram called HNDCTL. In actual implementation, this subprogram must be replaced by a suitable device driver.

The bit assignment is shown in Table 2. It will be noted that 1 will be used to denote the "on" state while 0 will be used to denote the "off" state. The subroutine HNDCTL contains sufficient logic to ensure that when both bits assigned to a given axis are on, they will be treated as both off (that is, no acceleration along, or rotation about, that axis) to conserve fuel usage. The main purpose of this subroutine is to examine the 12 bits from the hand controllers and return two vectors \mathbf{a} and \mathbf{w} where

$$\mathbf{a} = [a_1, a_2, a_3]^T \quad \text{and} \quad \mathbf{w} = [w_1, w_2, w_3]^T$$

whose meaning have been explained in the previous section. It is important to remember that both \mathbf{a} and \mathbf{w} are expressed in the OMV body frame.

Ideally, the hand controllers signals should be sensed and updated every minor cycle. But because of timing considerations they will be sensed once every major cycle, and it is explicitly assumed that the bit states do not change during the entire major cycle. This is not an unreasonable assumption, since one major cycle is 0.1 second, which is in the neighbourhood of the average reaction time of the human operator. Besides, the OMV does not have a fast response because of its large mass and low thrust levels.

The acceleration vector \mathbf{a} must be expressed in LVF before it

can be used in solving the equations of motion. In the OMV software, this is carried out as mentioned previously by :

- a) calculating the inverse of the direction cosine matrix C^{-1} ,
- b) transforming the vector a to A in LVF, and
- c) placing A in an acceleration matrix AA .

Step a) is carried out by a subroutine called DCSINV while steps b) and c) are carried out by subroutines DMUL and STORE in subroutine MOTION. At the end of the N minor cycles, the subroutine SOLVE is invoked to obtain solutions to the equations of motion numerically.

B) Numerical Solutions :

A three step Adam-Bashforth method (References 11-14) is used to obtain solutions to the equations of motion. This method is well known, and will not be elaborated here. Essentially, the set of three coupled second differential equations are re-written as a set of six simultaneous first order differential equations, and the solution computed. The six initial conditions needed for the computation are provided by the six components of the relative position and velocity vectors. Subroutine SOLVE takes the relative displacement and velocity vectors as initial conditions of the previous major cycle, and returns the new position and velocity vectors. A subroutine called STATE is then invoked to assemble the state vector.

C) Output Section :

A subroutine called OUTPUT is responsible for conveying information to the outside world. In normal operations, no output

is generally expected, but during testing, it is necessary to be able to monitor the progress of the simulation. At present, one can, via the use of flags, control the form and type of output. By way of example, one can request OMV to print a time sequence of state vectors at 1 second intervals on the printer, or display the position and orientation of the mobile base (on the flat floor) graphically, or disable all outputs altogether.

A fairly simple graphics package called PLOT is implemented to provide graphics output. This package is developed for the initial software checking only; namely to provide the operator with some form of visual output. It must be emphasized that this package is hardware dependent, and is not compatible with the PDP 11/34 mini-computer. The present graphics package runs on an IBM Personal Computer fitted with a TECMAR GRAPHICS MASTER board and an IBM monochrome monitor. A resolution of 640 by 352 is used for the package, although the system has a potential resolution of 720 by 700 pixels (Reference 15). PLOT uses escape codes to generate the top or side view of the mobile base (including the mock up module). A listing of this package, written in FORTRAN 77, is included in Appendix 3. It is anticipated that this package can be modified to run on the Evans and Sutherland color graphics terminal driven by a VAX 780.

The entire OMV module is written in FORTRAN 77, and all floating point computations are carried out in double precision. The usual structured programming technique is used. Modular design is faithfully adhered to, so that subroutines can be easily updated or replaced. At times, efficiency may be sacrificed for

code clarity, thereby making the code much easier to maintain and modify. During the design phase, flexibility is emphasized. Model parameters are inputted from disk files. Thus, modifications on the flat floor system will not involve any changes to the OMV source code. Appendix 4 shows the various data files used. Explanations for the various quantities are included as part of the record so that one can easily modify the configuration, initial conditions and so forth without having to refer to the source listing. A complete listing of OMV is included in Appendix 5, and a hierarchial chart is shown in Figure 4.

V) Testing and Results

Initial testing of the OMV software is conducted using an IBM Personal Computer with 8087 arithmetic co-processor. The same source code without the graphics option has been uploaded to the PDP 11/34 at MSFC and executed successfully.

The nature of the model is such that the major source of error would arise from the numerical solutions of the equations of motion. Thus, much effort has been spent to ensure that the Adam-Bashforth method yields accurate results. An error analysis of this method shows that the error is of the order of h^5 where h is the step size. In the present work, the step size is typically 0.01. This, coupled with the fact that all computations are carried out in double precision, means that the expected truncation error is of the order of 10^{-10} -- a figure that is too good to be true.

The following tests were conducted to verify that this

method does indeed give accurate solutions. The homogeneous case is first considered. Physically, this corresponds to the situation where the operator leaves all the controls in neutral so that

$$a = [0,0,0]^T \quad \text{and} \quad w = [0,0,0]^T$$

Thus, the equations of motion reduce to :

$$\begin{aligned}\ddot{X} &= -2\omega \dot{Z} \\ \ddot{Y} &= -\omega^2 Y \\ \ddot{Z} &= 2\omega \dot{X} + 3\omega^2 Z\end{aligned}$$

This set of equations can be solved numerically using the Adam-Bashforth method. Further, if X_1, X_2, X_3 and V_1, V_2, V_3 are the initial conditions, it can be shown that the analytical solutions are :

$$X(t) = X_1 - \frac{(3\Omega t - 4\sin\Omega t)}{\Omega} V_1 - 6(\Omega t - \sin\Omega t) X_3 - \frac{(1 - \cos\Omega t)}{\Omega} V_3$$

$$\dot{X}(t) = -(3 - 4\cos\Omega t) V_1 - 6\Omega(1 - \cos\Omega t) X_3 - 2(\sin\Omega t) V_3$$

$$Y(t) = (\cos\Omega t) X_2 + \left(\frac{\sin\Omega t}{\Omega}\right) V_2$$

$$\dot{Y}(t) = -\Omega(\sin\Omega t) X_2 + (\cos\Omega t) V_2$$

$$Z(t) = \frac{2(1 - \cos\Omega t)}{\Omega} V_1 + (4 - 3\cos\Omega t) X_3 + \frac{\sin\Omega t}{\Omega} V_3$$

$$\dot{Z}(t) = 2(\sin\Omega t) V_1 + 3\Omega(\sin\Omega t) X_3 + (\cos\Omega t) V_3$$

Thus, the numerical solutions can be compared directly with the analytical ones. Here, Ω is the orbital velocity, and for a circular orbit, Ω can be calculated :

$$\Omega = G M_e / (R_o + H)^3$$

where G is the universal gravitation constant, M_e is the mass of the earth, R_o is the mean earth radius and H is the altitude. Note that at higher orbits, Ω approaches 0 and the equations of motion approach

$$\begin{aligned}\ddot{X} &\rightarrow 0 \\ \ddot{Y} &\rightarrow 0 \\ \ddot{Z} &\rightarrow 0\end{aligned}$$

and better agreement between numerical and analytical results are expected for high altitudes than lower orbits. A computer program called ADAM has been developed that would, given a set of initial conditions, calculate both the numerical and analytical solutions to the equations of motion. The source listing of ADAM is shown in Appendix 5. In the present set of tests, an altitude of 200 kilometers ($\Omega = 0.00118$ rad/sec) is used throughout. This altitude represents the lowest design orbit of the Orbital Maneuvering Vehicle. Table 3 shows a comparison between the analytical and numerical solutions at this altitude, using the initial conditions :

$$\begin{aligned} X_1 &= 0, & X_2 &= X_3 = 0 \\ V_1 &= 0.05, & V_2 &= V_3 = 0 \end{aligned}$$

The result shows that the two solutions agree to better than 3×10^{-8} in 60 minutes, or about 0.03 milli-meters. This figure is well below the expected accuracy of the flat floor simulation system. This suprisingly small error comes from the fact that the angular velocity Ω is quite small. When $\Omega = 1.0$ is used, (this angular frequency does not make sense physically, as it represents an orbit well below the earth's surface, but consititutes a valid situation mathematically), the errors propagate quite fast as to render the comparison meaningless after 10 minutes.

A second test was carried out at the same altitude, using null initial conditions:

$$\begin{aligned} X_1 &= X_2 = X_3 = 0 \\ V_1 &= V_2 = V_3 = 0 \end{aligned}$$

The hand controller signals were chosen to yield a constant acceleration along the X-axis in the LVF, that is $a = [0.025, 0, 0]^T$, and the orientation of the OMV is chosen to be aligned to the LVF at $t = 0$. The result after 4 seconds of simulation is shown in Table 4. A plot of the revelant dynamic variables as a function of time is shown in Figure 5. The result shows that the model behaves exactly as expected; namely that an acceleration along the X-axis gives rise to a Z component, as dictated by orbital mechanics. If we ignore the Z contribution for the time being, one can estimate the value of X and \dot{X} using Newton's laws (this is not an invalid estimate as the time inter-

val is quite short compared with the period of rotation) to be $X = 0.2$ meters, and $X = 0.1$ meter/sec respectively. These figures compare very favourably with the numerical results at $t = 4$ seconds.

A very interesting test was conducted in which the OMV is made to execute a pure pitch motion. In this test, it is assumed that the OMV is originally at rest, the initial conditions being :

$$\begin{aligned} X_1 &= X_2 = X_3 = 0 \\ V_1 &= V_2 = V_3 = 0 \\ r &= p = y = 0 \end{aligned}$$

where r , p , y represent the roll, pitch and yaw respectively. A pure pitch motion would correspond to a rotation about the 2-axis. Mathematically,

$$r = y = 0, \text{ and } p = w_2 = 0$$

When the OMV is executed in this mode, the state vectors are fed into the SVX module, with the result that the state vector is translated into a sequence of commands CMD. This sequence of commands is to be transmitted to the flat floor. Table 5 shows the relevant commands for the mobile base. As verified by the graphics display, the mock up module mounted on the mobile base executes a pure pitch at the same rate as the OMV, while the mobile base has to translate along the +X direction. In addition, the pivot point is progressively lowered as expected. This test shows that the modules OMV and SVX are properly interfaced, and that correct results are produced. The command strings as out-

putted by the system to the flat floor is shown in Figure 6.

To further ascertain that the system is functioning properly, the hand controller signals corresponding to a translation along l-axis and a yaw is generated. The revelant commands to the flat floor system is shown in Table 6. A pictorial representation of the mobile base and mock up is as shown in Figure 7. Note that the path of the center of mass of the mock up exactly duplicates that of the OMV.

In summary, various tests conducted have shown that the OMV-SVX system functions properly. By way of example, a pure yaw motion of the OMV demands that the mobile base describes a circular path as shown in Figure 8. There is just one area that needs further investigation, namely timeing considerations. This system must be able to complete all the computation within 0.1 second -- a major cycle. When the system is uploaded to the PDP 11/34, it was discovered that the computer took more than 0.1 seconds to complete one major cycle of computation. At this juncture, one can take one of the following three corrective actions :

- a) Use a faster host computer (VAX 780)
- b) Use single precision computation, or
- c) Increase the step size in the numerical methods.

Of the three choices, the first method is clearly desirable, but until the VAX is installed, one must explore the remaining alternatives. Table 7 shows a time comparison between single and double precision arithmetic when the OMV is run until identical

parameters on the PDP 11/34 computer. The result shows little improvement in execution time. This is not surprising since the computer is equipped with hardware floating point capability. The only remaining recourse is to increase the step size, thereby reducing the number of steps (and hence the number of iterations). It is discovered that the numerical solution to the equations of motion took most of the computation time. Table 8 shows a similar time test for various steps N and retaining double precision arithmetic after the code has been suitably optimized. The data show that a step size of $h = 0.025$ seconds ($N = 4$) satisfies the time requirement. The price to be paid is that the error associated with the numerical process may increase. Table 9 shows a comparison test for $N = 10$ and $N = 4$ using the program ADAM. The result suggests that there is an optimum N somewhere between 4 and 10 in which the error is a minimum, but this question is not pursued any further. The result also shows that the error does not increase substantially over the same period of 60 minutes whether we use $N = 10$ or $N = 4$. Using $N = 4$, the deviation from the analytical solution is still much less than the accuracy of the flat floor system.

Conclusion

The series of tests conducted, some of which are not reported here, shows that the simplified mathematical of the Orbital Maneuvering Vehicle is functioning properly, and that it interfaces properly with the State Vector Transformation module SVX to produce correct sequences of commands to the flat floor.

By choosing a coarser step in the numerical integration process, OMV is able to complete all the necessary computation within a major cycle, without compromising on the accuracy. The final acid test cannot be conducted until the flat floor hardware is operational.

List of Tables

Table 1

OMV Mass Characteristics

<u>Dynamic Variable</u>	<u>Value</u>	<u>unit</u>
Mass M	3282.75	kg
I_{11}	7048.37	kg m ²
I_{22}	3713.95	kg m ²
I_{33}	3713.95	kg m ²

Table 2

Hand Controller Bit Assignments

<u>bit</u>	<u>Meaning</u>
1	Acceleration along +1 direction
2	Acceleration along -1 direction
3	Acceleration along +2 direction
4	Acceleration along -2 direction
5	Acceleration along +3 direction
6	Acceleration along -3 direction
7	+ roll; CCW rotation about 1-axis
8	- roll; CW rotation about 1-axis
9	+ pitch; CCW rotation about 2-axis
10	- pitch; CW rotation about 2-axis
11	+ yaw; CCW rotation about 3-axis
12	- yaw; CW rotation about 3-axis

Table 3

Comparison Between Analytical and Numerical Solutions

Time in Minutes	X (meters)		Z (meters)	
	Numerical	Analytical	Numerical	Analytical
0	0.000000	0.000000	0.000000	0.000000
5	13.746736	13.746736	5.271240	5.271240
10	20.161917	20.161917	20.427114	20.427114
15	12.828963	12.828962	43.576117	43.576178
20	-12.952950	-12.952952	71.829442	71.829444
25	-59.582227	-59.582233	101.660919	101.660923
30	-126.855533	-126.855544	129.347660	129.347664
35	-211.993176	-211.993191	151.434377	151.434380
40	-309.986003	-309.986022	165.164663	165.164664
45	-414.220544	-414.220565	168.824984	168.824985
50	-517.304365	-517.304388	161.958539	161.958536
55	-611.988644	-611.988666	145.422253	145.422248
60	-692.072815	-692.072834	121.279843	121.279843

Note : X and Z are expressed in Local Vertical Frame.

Table 4

OMV Acceleration Along +X Direction

Time in Seconds	X in meters	\dot{X} in meters	Z in meters	\dot{Z} in meters
.0	0.000000	0.000000	0.000000	0.000000
.5	0.002940	0.012125	0.000001	0.000007
1.0	0.012128	0.024625	0.000009	0.000029
1.5	0.027565	0.037125	0.000032	0.000065
2.0	0.049253	0.049625	0.000077	0.000117
2.5	0.077190	0.062125	0.000152	0.000183
3.0	0.111377	0.074624	0.000263	0.000264
3.5	0.151814	0.087124	0.000418	0.000360
4.0	0.198501	0.099624	0.000625	0.000471

Initial conditions :

$$X_1 = X_2 = X_3 = 0 \quad \text{and}$$

$$V_1 = V_2 = V_3 = 0$$

Note : All quantities are expressed in Local Vertical Frame.

Table 5

OMV -- Pure pitch motion at 0.017453 rad/sec

Time (Sec)	Pitch (Rad)	X (meters)	Z (meters)
0	0.0000	5.0000	2.4384
4	0.0698	5.0010	2.3852
8	0.1396	5.0074	2.3324
12	0.2094	5.0167	2.2800
16	0.2793	5.0295	2.2284
20	0.3491	5.0460	2.1778
24	0.4189	5.0659	2.1285

Note : All measurements are in flat floor coordinates.
Please see Appendix 1.

Table 6

Motion of the Mobile Base under
constant acceleration of $[0.025, 0, 0]^T$ and constant yaw at 0.08675 rad/sec

Time (Sec)	X (meters)	Y (meters)	Z (meters)	Yaw (rad)
0	0.0000	11.6680	2.4384	0.0000
4	0.2752	11.2418	2.4390	0.3470
8	1.0709	11.0039	2.4433	0.6940
12	2.2919	11.1199	2.4545	1.0410
16	3.7925	11.7135	2.4750	1.3880
20	5.3934	12.8512	2.5062	1.7350
24	6.9035	14.5350	2.5480	2.0820

Note : X, Y and Z are expressed in flat floor coordinates. Please see Appendix 1.

Table 7

OMV Time Test

No of Steps	Average execution time per major cycle	
	Single Precision	Double Precision
4	0.077	0.084
5	0.090	0.099
6	0.103	0.113
7	0.117	0.128
8	0.130	0.143
9	0.144	0.158
10	0.157	0.173

Table 8

Optimized OMV Execution Times Per Major Cycle
As A Function Of Number Of Steps N

N	Execution time (Sec)
4	0.068
5	0.079
6	0.090
7	0.100
8	0.111
9	0.122
10	0.132

Table 9

Comparison Test Between $N = 4$ and $N = 10$ Steps

Time in Minutes	Solution		
	Analytic	Numeric	
		$N = 10$	$N = 4$
0	0.000000	0.000000	0.000000
5	13.746736	13.746736	13.746736
10	20.161917	20.161917	20.161917
15	12.828962	12.828963	12.828961
20	-12.952953	-12.952950	-12.952956
25	-59.582233	-59.582227	-59.582237
30	-126.855544	-126.855533	-126.855551
35	-211.993191	-211.993176	-211.993200
40	-309.986022	-309.986003	-309.986034
45	-414.220565	-414.220544	-414.220579
50	-517.304388	-517.304365	-517.304403
55	-611.988666	-611.988644	-611.988681
60	-692.072834	-692.072815	-692.072847

List of Figures

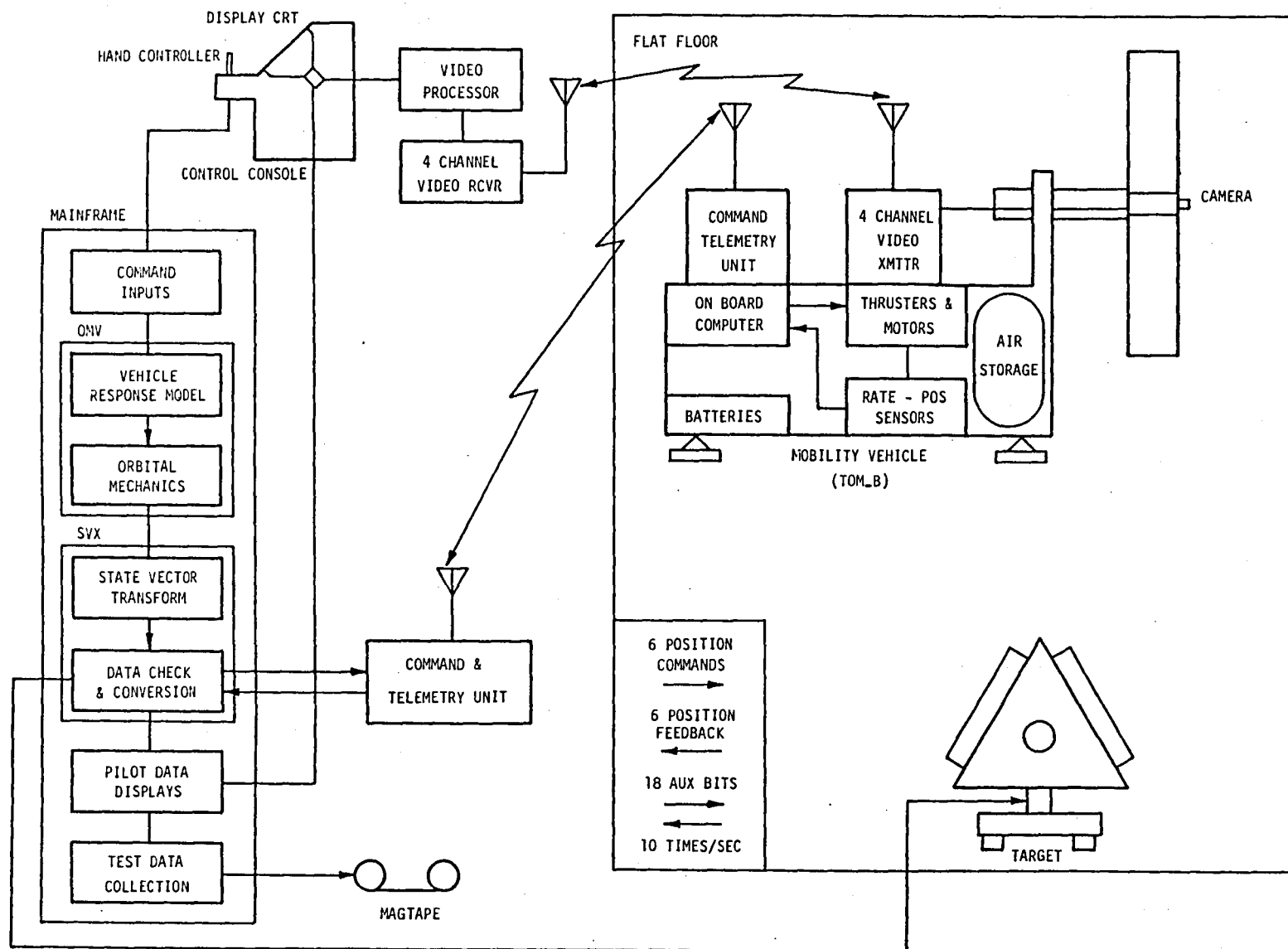


Figure 1. MSFC Flatfloor Simulation System

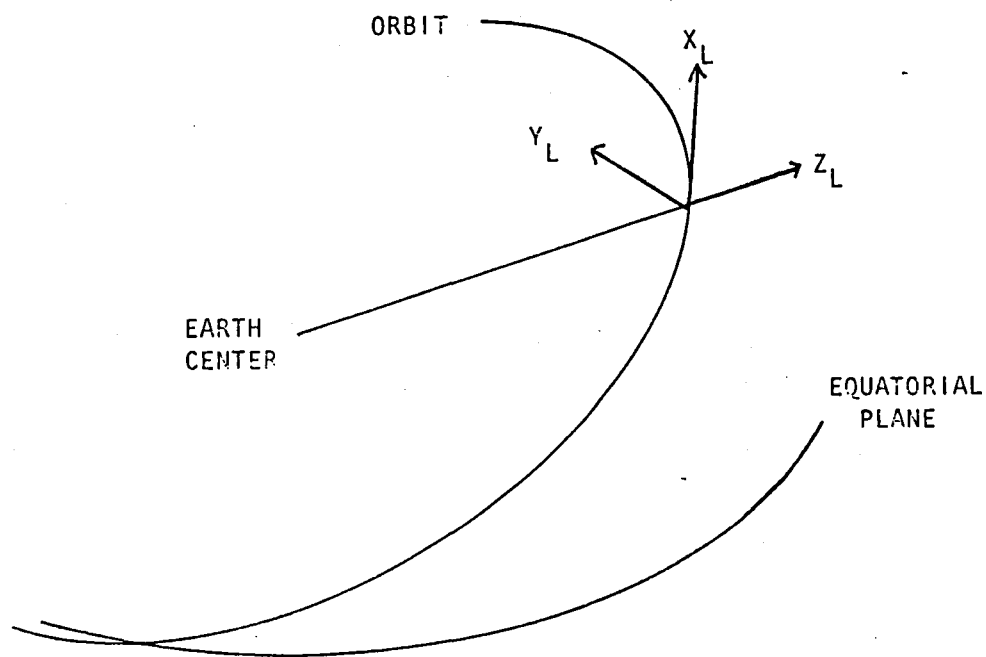
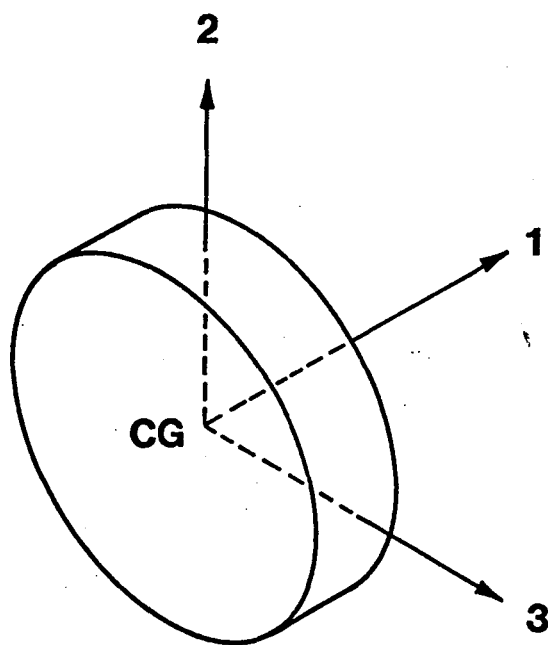


Fig.2 Local Vertical Frame (L)

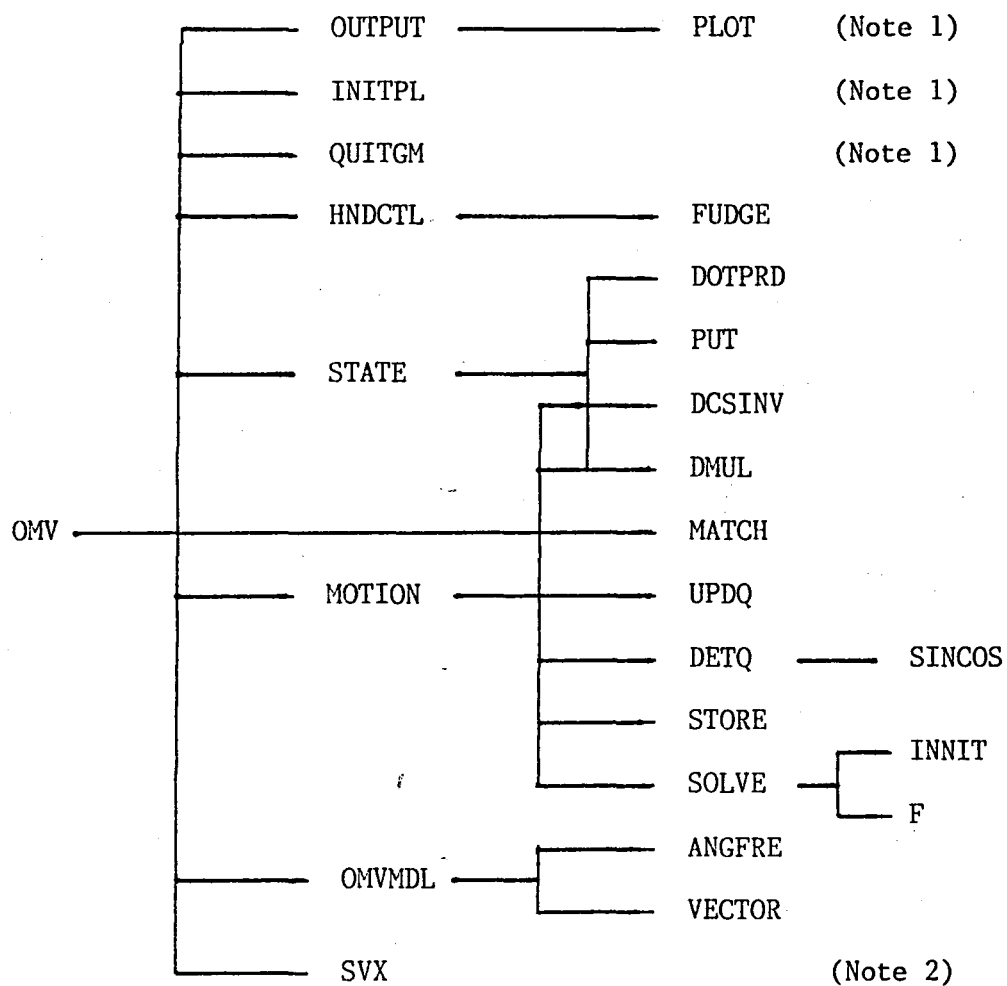


OMV Body Frame

Fig 3

Figure 4

OMV Heirarchial Chart



Note 1 : Hardware incompatible graphics package.

Note 2 : Vector Transformation Module. See Reference 1.

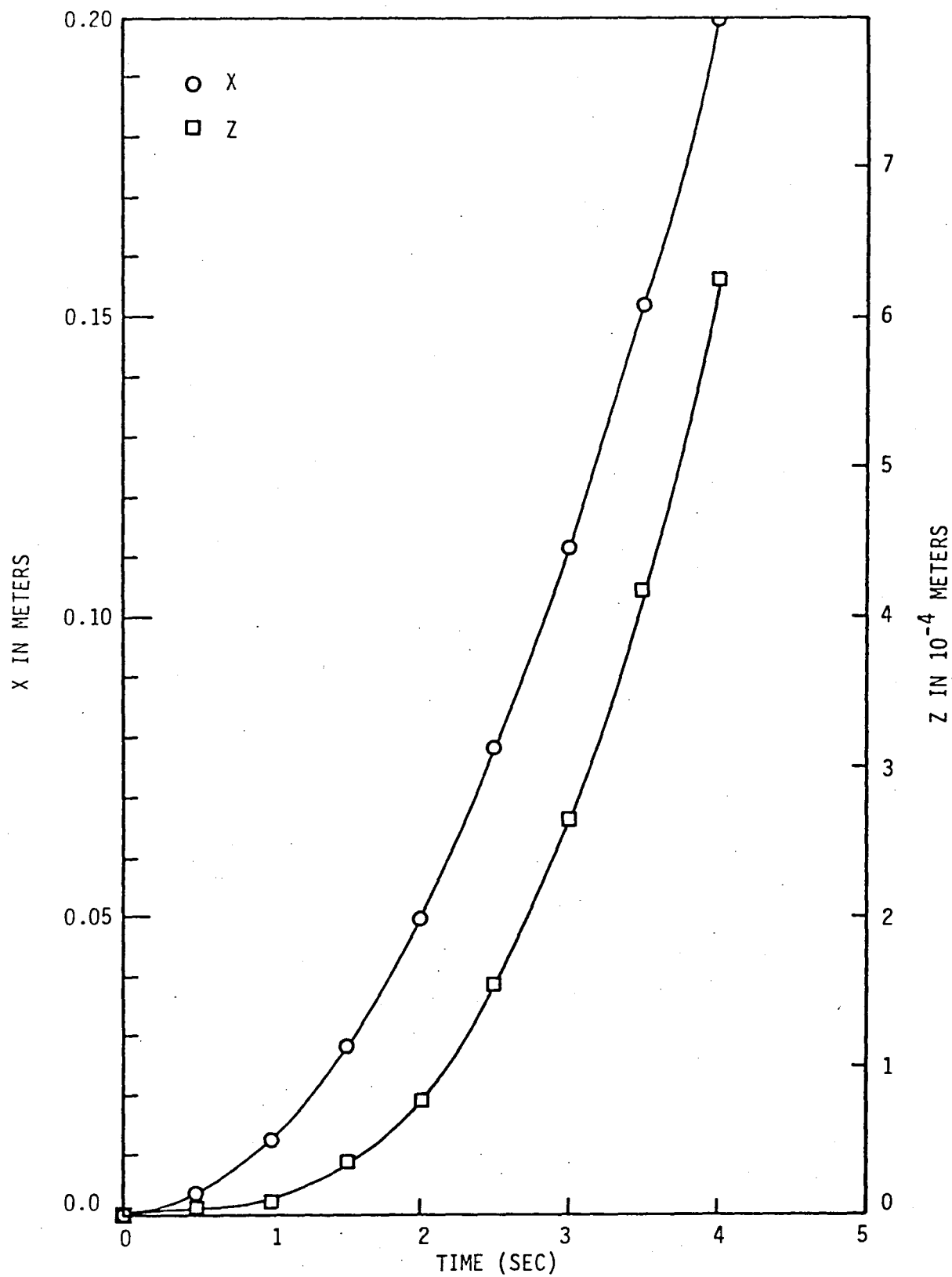


Figure 5. Translation Along X-Axis

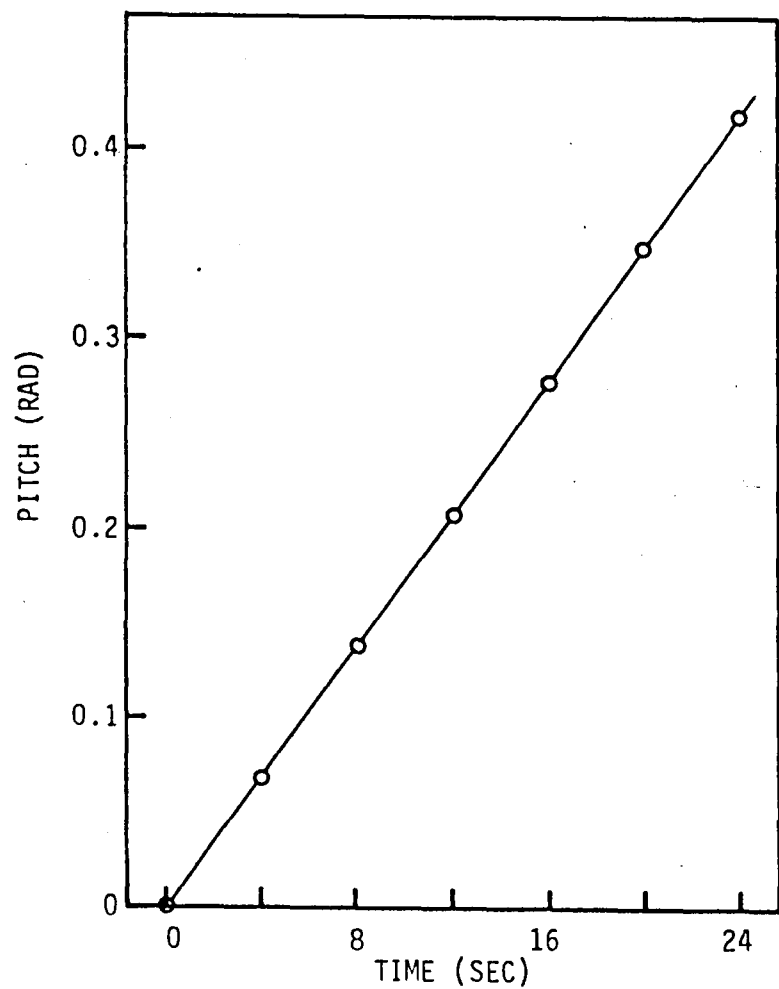
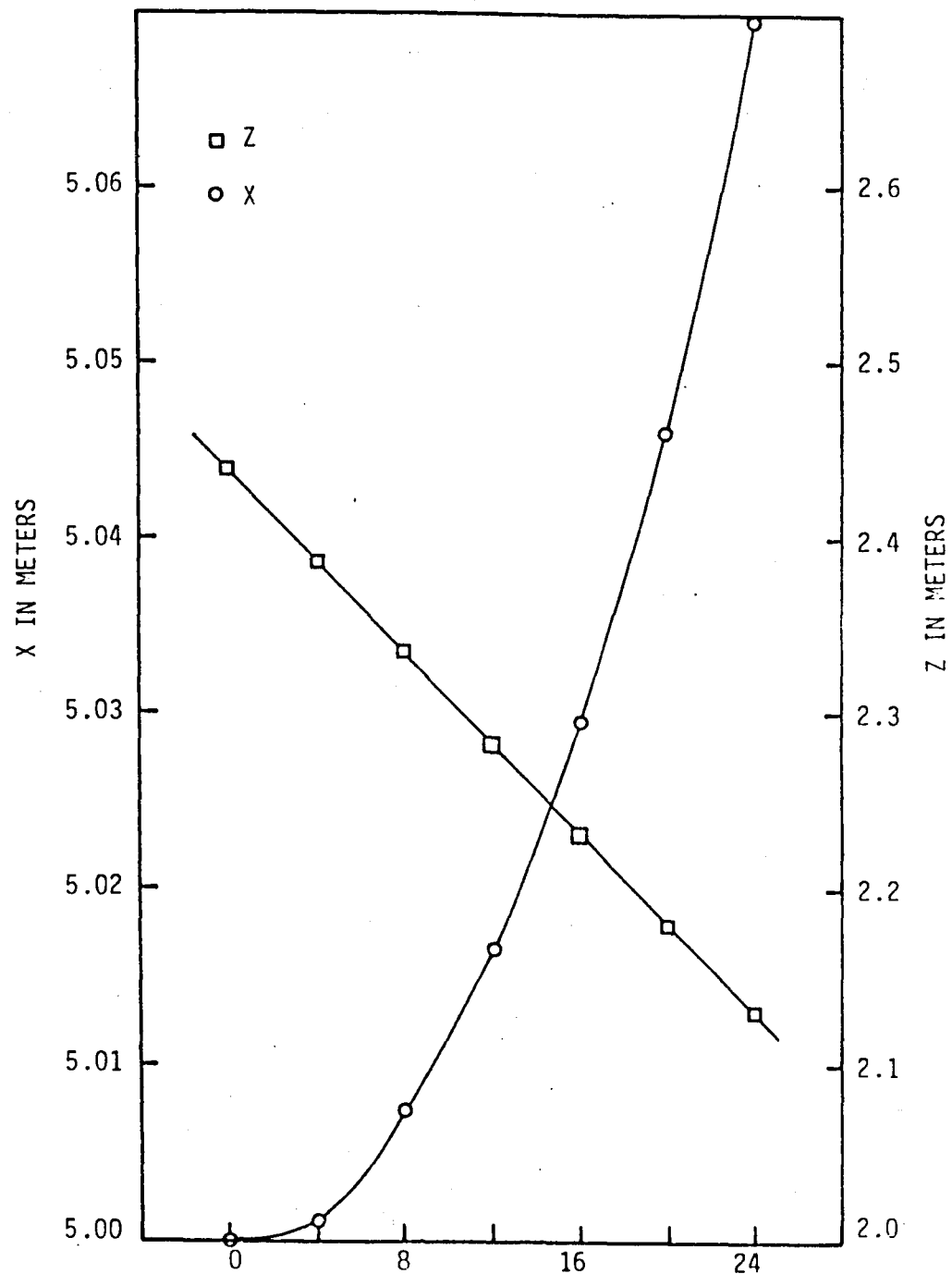


Figure 6. Pure Pitch Motion
at 0.017453 rad/sec.



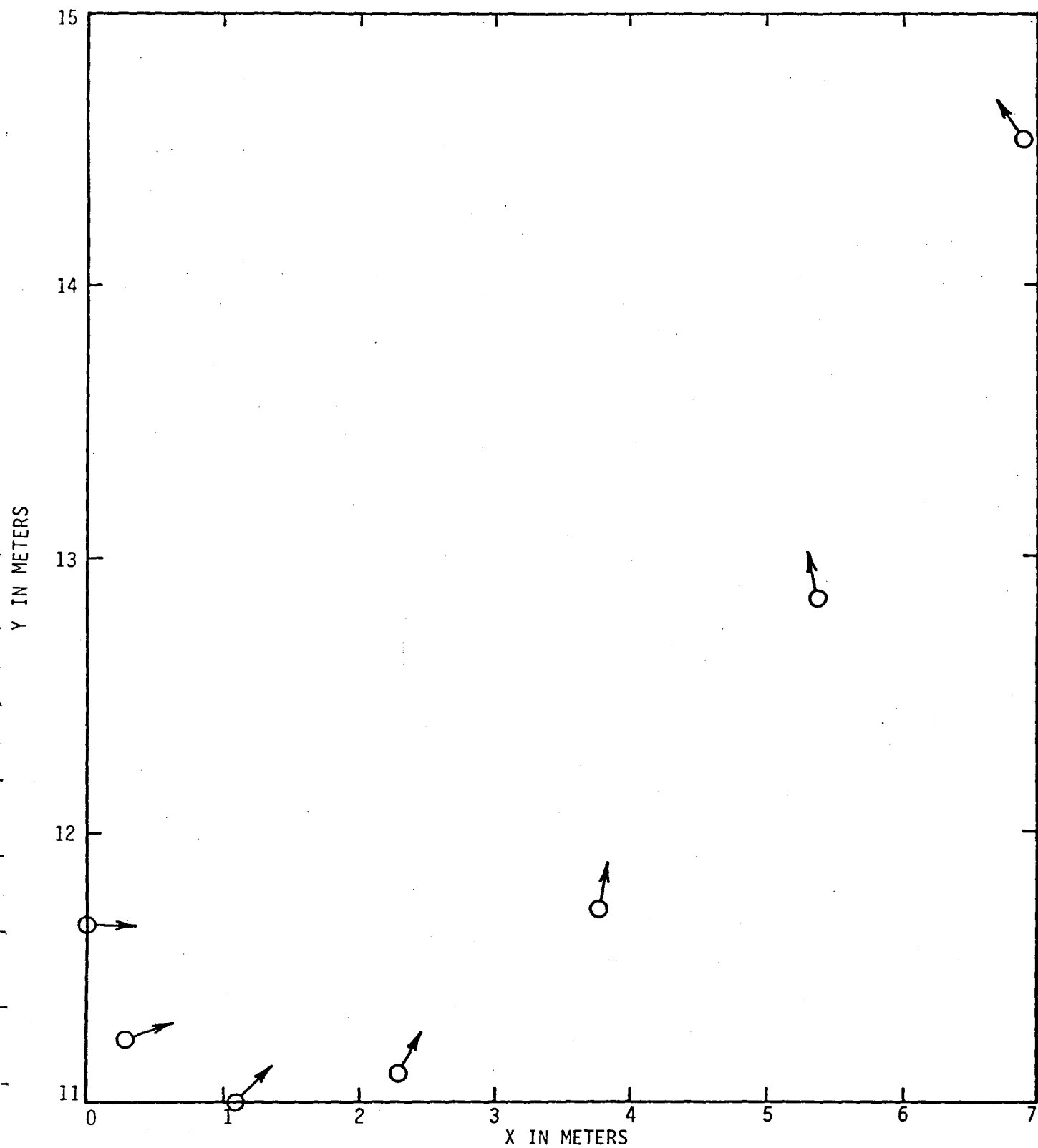


Figure 7. Trajectory of Mobile Base When
OMV is Executing a Translation & Yaw

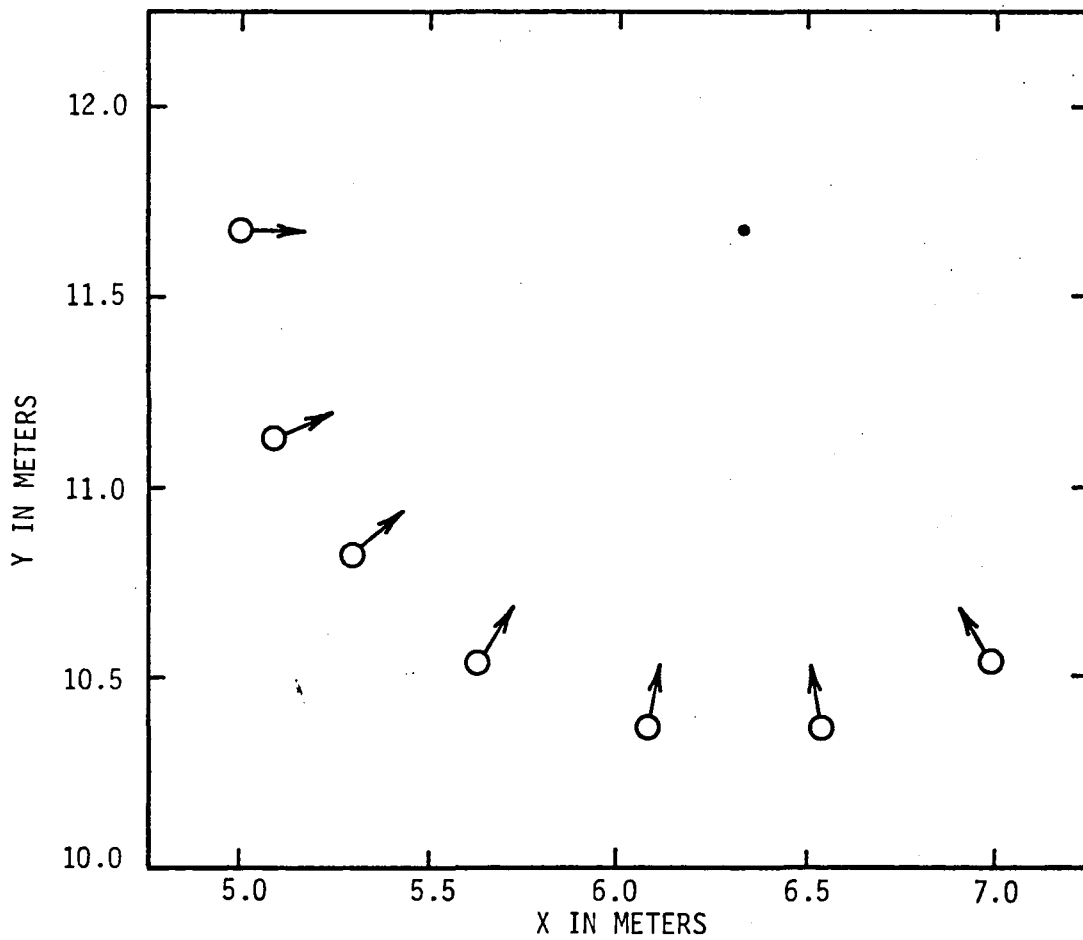


Figure 8. OMV Pure Yaw Motion

List of References

List of References

1. "State Vector Transformation Module", W. Teoh, NASA Technical Report, May 9, 1984. Contract # NAS8-35670.
2. "Teleoperator Maneuvering System Preliminary Definition Study", prepared by Program Development, MSFC, June 1983.
3. "Classical Mechanics", H. Goldstein, Addison-Wesley Inc, 1965.
4. "Mechanics" K. Symon, Addison-Westley Inc, 1964.
5. "Equations of Motion for Six Degrees Of Freedom TRS Simulation", J. Galaboff, NASA MSFC System Dynamics Laboratory Memo ED15-78-34, August, 1978.
6. "On The Use Of Quaternions In Simulation Of Rigid Body Motion", A. C. Robinson, Wright Air Development Center, 1958.
7. "A New Method For Performing Digital Control System Attitude Computation Using Quaternions", B. P. Ickes, AIAA Journal 8(1970)13.
8. "Design Of A Terminal Pointer Hand Controller For Teleoperator Applications:", E. L. Saenger and W. S. Woltosz, URS/Matrix Company Final Report, NASA Contract # NAS8-28760, 1973.
9. "Quaternions For Control Of Space Vehicles", A. C. Hendley, Sperry Rand Corp, NASA Contract # NAS8-20055, 1973.
10. "Software Specifications For Docking Simulations Of The Orbital Maneuvering Vehicle (OMV)", J. D. Micheal, NASAN. MSFC System Dynamics Lab, Memo ED15-83-64, 1984.
11. "Discrete Variable Methods In Ordinary Differential Equations", P. Henrici, John Wiley & Sons, 1962.
12. "Numerical Methods With FORTRAN IV", D. McCracken, John Wiley & Son, 1972.
13. "Mathematical Modelling And Digital Simulation For Engineerers And Scientists", J. Smith, John Wiley & Son, 1977.
14. "Applied Numerical Methods", B. Carnahan, H. Luther and J. Wilkes, John Wiley & Son, 1969.
15. "GRAPHICS MASTER Reference Mannual", Tecmar Corp, 1983.

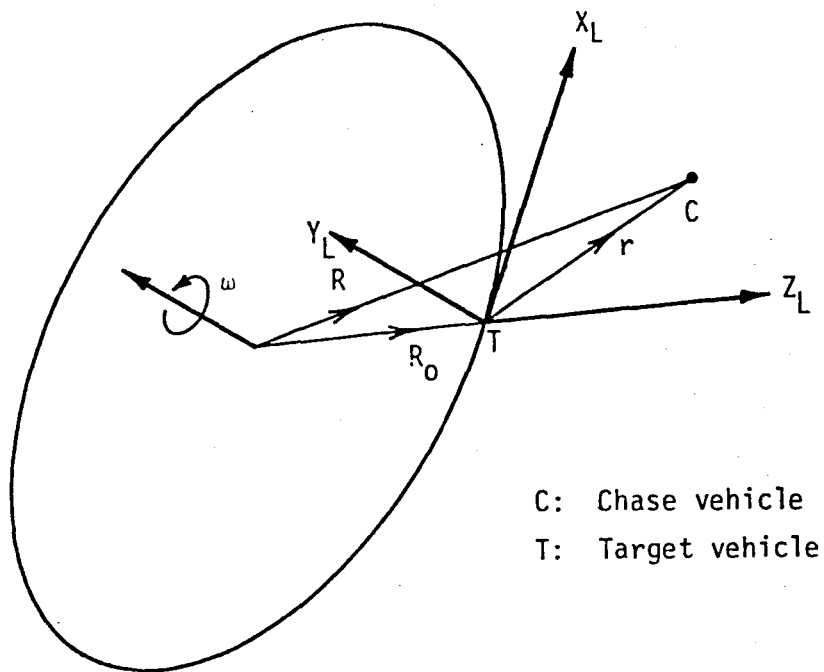
Appendix 1

OMV Translational Equations of Motion

Appendix 1.

OMV Translational Equations Of Motion

Consider a target vehicle orbiting the earth with an angular velocity ω and an orbit radius of R_0 . We can define a local vertical frame (LVF) at the center of gravity of this vehicle as shown in the figure below :



Here, X_L , Y_L and Z_L are the three orthogonal axes of the LVF. We can imagine that the center of the earth may be considered as the origin of the inertial coordinate frame. We can choose the axes of this coordinate system as shown. In particular, Y_E is parallel to Y_L . We shall use the subscript L to denote those quantities that are expressed in the LVF, while the subscript E shall be used for those quantities expressed in the inertial frame. The point C in the above figure represents the center of mass of the chase vehicle (OMV)

The equation of motion of the chase vehicle is easily deduced from Newton's second law, namely,

$$M_c \ddot{\mathbf{R}} = \mathbf{F}_g + \mathbf{F}_c \quad (1)$$

This equation is written in the inertial frame. Here, M_c is the mass of the chase vehicle, \mathbf{F}_g is the gravitational force exerted on the vehicle by the earth, and \mathbf{F}_c is the control force exerted on the vehicle from the on-board thrusters and jets. The objective of this exercise is to derive the equation of motion in terms of \mathbf{r} and its time derivatives. Namely, we wish to express the motion of the chase vehicle (OMV) in local vertical frame. This choice turns out to be very convenient for docking maneuvers.

From the above figure, it is obvious that

$$\mathbf{R} = \mathbf{R}_o + \mathbf{r}_E \quad (2)$$

it follows that

$$\ddot{\mathbf{R}} = \ddot{\mathbf{R}}_o + \ddot{\mathbf{r}}_E \quad (3)$$

Since the LVF is a rotating frame, we can use the operator :

$$\{ d/dt \}_E = \{ d/dt + \mathbf{w} \times \}_L$$

Applying this operator to \mathbf{r} twice, we have

$$\dot{\mathbf{r}}_E = \dot{\mathbf{r}}_L + \mathbf{w} \times \mathbf{r}_L$$

and

$$\begin{aligned} \ddot{\mathbf{r}}_E &= d/dt (\dot{\mathbf{r}}_L + \mathbf{w} \times \mathbf{r}_L) + \mathbf{w} \times (\dot{\mathbf{r}}_L + \mathbf{w} \times \mathbf{r}_L) \\ &= \ddot{\mathbf{r}}_L + \mathbf{w} \times \dot{\mathbf{r}}_L + \mathbf{w} \times \dot{\mathbf{r}}_L + \mathbf{w} \times (\mathbf{w} \times \mathbf{r}_L) \\ &= \ddot{\mathbf{r}}_L + 2\mathbf{w} \times \dot{\mathbf{r}}_L + \mathbf{w} \times (\mathbf{w} \times \mathbf{r}_L) \end{aligned} \quad (4)$$

From equations (3) and (4), we have :

$$\begin{aligned}\ddot{\mathbf{R}} &= \ddot{\mathbf{R}}_O + \ddot{\mathbf{r}}_E \\ &= \ddot{\mathbf{R}}_O + \ddot{\mathbf{r}}_L + 2\mathbf{w} \times \dot{\mathbf{r}}_L + \mathbf{w} \times (\mathbf{w} \times \mathbf{r}_L)\end{aligned}$$

Furthermore, for a circular orbit,

$$\ddot{\mathbf{R}}_O + w^2 \mathbf{R}_O = 0$$

therefore,

$$\ddot{\mathbf{R}} = -w^2 \mathbf{R}_O + \ddot{\mathbf{r}}_L + 2\mathbf{w} \times \dot{\mathbf{r}}_L + \mathbf{w} \times (\mathbf{w} \times \mathbf{r}_L) \quad (5)$$

It is clear at this point that the equations of motion (1) can be rewritten in terms of \mathbf{r}_L and \mathbf{R}_O and their time derivatives. Thus the subscript will be dropped from here on. Recall that

$$\begin{aligned}\mathbf{R} &= \mathbf{R}_O + \mathbf{r} \\ R^2 &= (\mathbf{R}_O + \mathbf{r}) \cdot (\mathbf{R}_O + \mathbf{r}) \\ &= R_O^2 + r^2 + 2\mathbf{R}_O \cdot \mathbf{r} \\ &= R_O^2 + 2\mathbf{R}_O \cdot \mathbf{r} \\ &= R_O^2 \{ 1 + 2(\mathbf{R}_O \cdot \mathbf{r}) / R_O^2 \}\end{aligned}$$

so that

$$\begin{aligned}R^{-3} &= R_O^{-3} \{ 1 + 2(\mathbf{R}_O \cdot \mathbf{r}) / R_O^2 \}^{-3/2} \\ &\cong R_O^{-3} \{ 1 - 3(\mathbf{R}_O \cdot \mathbf{r}) / R_O^2 \}\end{aligned}$$

Thus,

$$\begin{aligned}\mathbf{F}_g &= -(GM_e M_c / R^3) \mathbf{R} \\ &= -(GM_e M_c / R_O^3) (\mathbf{R}_O + \mathbf{r}) (1 - 3(\mathbf{R}_O \cdot \mathbf{r}) / R_O^2) \\ &= -w^2 M_c (\mathbf{R}_O + \mathbf{r}) (1 - 3(\mathbf{R}_O \cdot \mathbf{r}) / R_O^2) \\ &\cong -w^2 M_c (\mathbf{R}_O + \mathbf{r} - 3(\mathbf{R}_O \cdot \mathbf{r}) / R_O^2) \mathbf{R}_O\end{aligned} \quad (6)$$

since for a circular orbit, $w^2 = GM_e / R_O^3$. Substituting equations (5) and (6)

into (1), we have :

$$M_c \{-w^2 R_o + \ddot{r} + 2w \times \dot{r} + w \times (w \times r)\} = F - M_c w^2 \{R_o + r - 3(R_o \cdot r)/R_o^2\}$$

If we define $A = F_c / M_c$, then we have :

$$-w^2 R_o + \ddot{r} + 2w \times \dot{r} + w \times (w \times r) = A - w^2 R_o - w^2 r + 3w^2 (R_o \cdot r / R_o^2) R_o$$

which, after re-arranging, gives :

$$\ddot{r} = A - 2w \times \dot{r} - w^2 r - w \times (w \times r) + 3w^2 (R_o \cdot r / R_o^2) R_o \quad (7)$$

Now, we shall state r , R_o and w in cartesian coordinates. It is explicitly assumed that the unit vectors i , j and k are directed along X_L , Y_L and Z_L axes respectively. Thus,

$$\begin{aligned} r &= [X, Y, Z]^T \\ R_o &= [0, 0, R_o]^T \\ w &= [0, w, 0]^T \quad \text{and} \\ A &= [A_x, A_y, A_z]^T \end{aligned}$$

and it can easily be shown that :

$$\begin{aligned} 2w \times \dot{r} &= [2w\dot{Z}, 0, -2w\dot{X}]^T \\ w \times (w \times r) &= [-w^2 X, 0, -w^2 Z]^T \\ 3w(R_o \cdot r / R_o^2) R_o &= [0, 0, 3w^2 Z]^T \quad \text{and} \\ w^2 r &= [w^2 X, w^2 Y, w^2 Z]^T \end{aligned}$$

and substituting into equation (7) yields

$$\begin{aligned} [X, Y, Z]^T &= [-2w\dot{Z}, 0, 2w\dot{X}]^T + [w^2 X, 0, w^2 Z]^T \\ &+ [-w^2 X, -w^2 Y, -w^2 Z]^T + [0, 0, 3w^2 Z]^T \\ &+ [A_x, A_y, A_z]^T \end{aligned}$$

or

$$\begin{aligned}\ddot{X} &= A_x - 2w\dot{Z} \\ \ddot{Y} &= A_y - w^2Y \\ \ddot{Z} &= A_z + 2w\dot{X} + 3w^2Z\end{aligned}\tag{8}$$

Equation (8) is the equation of motion of the chase vehicle relative to the target vehicle in local vertical frame.

Appendix 2

State Vector Transformation Module SVX

Technical Report

State Vector Transformation Module

Technical Report

Prepared for
George C. Marshall Space Flight Center
Marshall Space Flight Center
Huntsville, AL 35812

by

Dr. William Teoh
Kenneth Johnson Energy & Environment Center
University of Alabama in Huntsville
Huntsville, AL 35899

Date prepared : May 9, 1984

Contract # NAS8-35670

INTRODUCTION

The State Vector Transformation Module (SVX) is an interface between the OMV simulation model and the mobile base (TOM_B) of the flat floor simulation system. We can imagine the OMV simulation to be a free flying vehicle in space under human operator control, and at any particular instant, its state can be summarized as a fourteen-component vector called the state vector S. SVX takes this state vector as an input and generates an appropriate string of commands that is transmitted to TOM_B with the stipulation that if TOM_B executes this command string exactly, then the mock-up module mounted on TOM_B will exactly replicate the motion of the OMV as perceived by the operator.

References 1), 2) and 3) are reports that pertain to the various aspects of the OMV. From these reports, the various components that make up the state vector can be deduced and are presented below :

<u>Component</u>	<u>Symbol</u>	<u>Meaning</u>
1	X	position of the target vehicle relative to the OMV in local vertical frame LVF.
2	Y	
3	Z	
4	V_x	relative velocity of the chase vehicle in LVF
5	V_y	
6	V_z	
7	L_x	angular momentum vector in LVF
8	L_y	
9	L_z	
10	q_1	attitude quaternions in body frame
11	q_2	
12	q_3	

13	q_4	
14	m	mass of OMV

It is often more convenient to consider the state vector to be made up of the following four vectors : $X = [X, Y, Z]^T$, $V = [V_x, V_y, V_z]^T$, $L = [L_x, L_y, L_z]$ and the unit quaternion $q = [q_1, q_2, q_3, q_4]^T$.

As mentioned earlier, the required command string must be derived from this state vector, and is transmitted to TOM_B as seven 16-bit words. The last word can either be a zero or a one, which is interpreted by the TOM_B Executive as rate or position control respectively. A brief explanation of the command string is shown below :

Component	Position control		rate control		coord. system
	symbol	meaning	symbol	meaning	
1	y	yaw of TOM_B	y	yaw rate	body frame
2	X	position of	V_x	velocity of	LVF
3	Y	TOM_B	V_y	TOM_B	
4	Z	pos of pivot	V_z	vel of pivot	
5	p	pitch angle	p	pitch rate	body frame
6	r	roll angle	r	roll rate	
7	1	pos. control	0	rate control	

Before the detailed analysis is presented, it is necessary to define the various coordinate systems used.

COORDINATE SYSTEMS

Several coordinate systems are used in this software module. Specifically, motion of the OMV is described in Local Vertical Frame (LVF) while the orienta-

tion of the OMV is described in body frame. Similarly, the position and velocity of the mobile base TOM_B is described in floor coordinates while the orientation of the mock-up module and TOM_B are described by their respective body frames.

A) the Local Vertical Frame (LVF)

Imagine a circular orbit that is inclined at an angle i with respect to the equatorial plane. A Local Vertical Frame is a non-stationary frame that has its origin at a point on this orbit such that

- (i) its Z_L axis is directed away from the earth's center,
- (ii) its X_L axis is directed tangential to the orbit and is perpendicular to its Z_L axis, and
- (iii) the Y_L axis is directed parallel to the angular momentum vector, as shown in Figure 1.

A subscript L will be used to indicated quantities defined in this coordinate system.

B) the Floor Coordinates (F)

The floor coordinates has its origin at one corner of the flat floor as shown in Figure 2. Its X_F axis is directed along the width of the floor, while the Y_F axis is directed along the length of the floor. Naturally, Z_F axis is directed vertically up.

C) the TOM_B body Frame (B)

This coordinate system is fixed with respect to the mobile base, and has its origin at the center of mass of the mobile base. Its X_B axis is directed towards the front of TOM_B, while its Z_B axis is parallel to the Z_F axis of the flat floor. A third axis Y_B is chosen so as to form an orthogonal right-handed

coordinate system, a top view of which is shown in Figure 3.

D) the Mock-Up Module Body Frame (M)

We shall assume that the mock-up module resembles the OMV in shape (that is, not unlike a pancake). The origin of its body frame coincides with its center of mass, and the X_M axis is directed towards the front of the module. Initially, at the start of the simulation, the Z_M axis is chosen to be parallel to Z_F , and the appropriate orthogonal axis is chosen as its Y_M axis, as indicated in Figure 4.

ANALYSIS

From the references mentioned above, it is obvious that the relative position and attitude from the state vector are relative quantities. Thus, initial conditions at the start of the simulation must be known. Figures 5 a) and b) shows the initial state of the mobile base and mock-up module at the start of the simulation. The quantities a, c, l, h and o can be obtained from measurement.

A necessary initial condition is that the operator must leave the hand controllers in the neutral position for at least one second so that the initial position of the OMV $[X_o, Y_o, Z_o]^T$ can be obtained. It is also assumed that the initial orientations of both the OMV and mock-up module are set in their home position. If the notation r, p, and y is used to indicate the roll, pitch and yaw of both the OMV and the mock-up, then,

$$[r_{OMV}, p_{OMV}, y_{OMV}]^T = [r_M, p_M, y_M]^T = [0, 0, 0]^T$$

It is obvious that the corresponding axes of the coordinate frames M, B and F are all parallel at this point in time. At any later time, the position of the OMV can be calculated from the state vector :

$$\begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} \quad [0]$$

Here, S_1 , S_2 , and S_3 are the first three components of the state vector. This position is measured relative to the starting point in the beginning of the simulation, and can be transformed to the position of the mock-up module in floor coordinates using the equation :

$$\begin{bmatrix} X_M \\ Y_M \\ Z_M \end{bmatrix} = \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} + \begin{bmatrix} c + 1 - X_o \\ a - Y_o \\ h - Z_o \end{bmatrix} \quad [I]$$

Equation [I] governs the transformation of the position vector of the OMV in LVF to a position vector for the mock-up module in floor coordinates, based on the initial conditions and the first three components of the state vector. Given that the instantaneous orientation of the module is $[r_M, p_M, r_M]^T$ as shown in Figure 6 a) and b), the position of TOM_B $[X_F, Y_F, Z_F]^T$ in floor coordinates is given by :

$$\begin{bmatrix} X_F \\ Y_F \\ Z_F \end{bmatrix} = \begin{bmatrix} X_M - (c + l\cos(p))\cos(y) \\ Y_M - (c + l\cos(p))\sin(y) \\ \delta \end{bmatrix} \quad [II]$$

Note that Z_F is the height of the center of mass of TOM_B from the floor (a constant quantity), and is not of interest here. In stead, the quantity of interest is Z , which is the height of the pivot point from the floor as shown in Figure 6, and

$$Z = Z_M - l\sin(p) \quad [III]$$

It follows that the velocity of TOM_B and the pivot point is given by

$$\begin{bmatrix} X_F \\ Y_F \\ Z \end{bmatrix} = \begin{bmatrix} X_M + (c + l \cos(p)) \sin(p)y + l \sin(p) \cos(y)p \\ Y_M - (c + l \cos(p)) \cos(p)y + l \sin(p) \sin(y)p \\ Z_M - l \cos(p)p \end{bmatrix} \quad [IV]$$

The above transformations take care of the position and velocity quantities.

The quaternions q_1, q_2, q_3, q_4 from the state vector specifies the OMV's attitude in body frame, as discussed in References 4) and 5). At any instant, its orientation is given by (see Ref 4) :

$$[r, p, y]^T = \alpha [0_x, 0_y, 0_z]^T \quad [V]$$

where

$$\alpha = 2 \cos^{-1}(q_4)$$

$$[0_x, 0_y, 0_z]^T = (iq_1 + jq_2 + kq_3) / (q_1 + q_2 + q_3)^{0.5} \quad [VI]$$

while their rates are $\mathbf{w}_B = [w_1, w_2, w_3]^T$ which can be calculated in the following manner:

Since the angular momentum vector $\mathbf{L} = [L_x, L_y, L_z]^T$ from the state vector is expressed in LVF, it is necessary to transform it to body frame using the equation :

$$\mathbf{L}_B = \mathbf{A} \mathbf{L} \quad [VII]$$

here \mathbf{A} is the direction cosine matrix which can be constructed from the attitude quaternions q_1, q_2, q_3 , and q_4

$$\mathbf{A} = \begin{bmatrix} q_4 + q_1 - q_2 - q_3 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & q_4 - q_1 + q_2 - q_3 & 2(q_2q_3 + q_3q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & q_4 - q_1 - q_2 + q_3 \end{bmatrix} \quad [VIII]$$

Knowing the moment of inertia tensor I , one can calculate the angular rates

$$\begin{aligned} w_B &= [w_1, w_2, w_3]^T \\ &= I^{-1} L_B = I^{-1} (A L) \end{aligned} \quad [IX]$$

Thus, one has all the needed information from the state vector to yield the necessary position or rate control commands.

ALGORITHM

The algorithm for SVX makes use of all the transformations described in the above section. Essentially, the algorithm uses the state vector and depending on the value of MODE, generates the appropriate command string CMDRAW.

Case 1 MODE \neq 0 (position control)

In this case, both orientation and position of the OMV are updated. A transformation is made to yield the position of the center of mass of TOM_B using equation [I] through [III]. The orientation of the mock-up module is obtained using equation [VI]. Using the previous notation, a seven element vector

$$[y, x_B, y_B, z, p, r, 1]^T$$

is generated. Each element of this vector is suitably scaled and round off to the nearest integer (16-bit word) and is the sole output of the SVX module. Rate information is not of interest when the system is in position control, and is therefore not transmitted. Throughout this module, the scale factors for all angular and displacement quantities are 10^4 and 10^3 respectively.

Case 2 MODE = 0 (rate control)

In this rate control mode, it is still necessary to update the orientation (equation [VI]) although it is no longer necessary to update the position of the OMV. The velocity of TOM_B in floor coordinates is determined from equation [IV] while the rates for roll, pitch and yaw are determined using equations [VII] through [X]. The seven 16-bit word command string is

$$[y, X_B, Y_B, Z, p, r, 0]^T$$

As before, each component of this vector is similarly scaled and rounded before returning.

Case 3 MODE <> 0 AND MODE <> 1

In this case, MODE is set to 1, and position control is assumed.

IMPLEMENTATION

This algorithm is implemented as a subroutine named SVX (S, CMDRAW, MODE) where the three items on the parameter list are the state vector output command string and control mode respectively.

The subroutine is implemented in FORTRAN 77, and the usual programming practices are adhered to. Most of the major steps are either properly documented in the form of COMMENT statements or implemented as subprograms, following a modular design approach. Whenever possible, structured codes are used unless severe degradation of execution speed may result.

SVX is compiled and tested using an IBM Personal Computer, and the source code, on completion of the testing, is uploaded to the PDP 11/34 computer at MSFC. Appendix I shows a complete listing of this module. A more detailed description of the testing procedure will be presented later in this section.

A local counter (COUNT) is initialized at load time, and updated during execution to enable SVX to determine the initial state on start up. During this period, other tasks are carried out as an integral part of the initialization process. This includes reading a file (SVXINT.DAT) for the values of c, l, a, h and o, as well as the inverse of the moment of inertia tensor I^{-1} .

This module assumes that the operator will, at start up, leave the hand controller at a neutral position for at least a second. During this interval, the initial state of the OMV is recorded, and the vector E where

$$\begin{aligned} E &= [E_1, E_2, E_3]^T \\ &= [c + l - X_0, a - Y_0, h - Z_0]^T \end{aligned}$$

is calculated. The roll, pitch and yaw of both the OMV and the mock-up module are initialized to zero during this process by invoking subroutine ZERO.

Subsequent calls to SVX causes a seven 16-bit command string in an INTEGER array called CMDRAW to be produced. Computation here depends on the value of MODE.

When MODE is non-zero, position control is assumed. SVX invokes subroutines QTRPY and UPDPOS to calculate the desired orientation and position of the OMV. A transformation is then made to determine the required position (of the mobile base TOM_B in floor coordinates) and orientation (of the mock-up module in body frame). Since the value of MODE cannot be changed in the course of a simulation, no rate information is calculated or retained.

When MODE is zero, rate control is used. First, QTRPY is called to calculate the orientation of the OMV; its position is not computed because it is not of interest while in rate control mode. The direction cosine matrix A is formed by invoking subroutine DIRCOS, and a simple matrix multiplication transforms

the angular momentum to body frame. Finally, the velocity of the OMV (from the state vector) is suitably transformed to yield the velocity of TOM_B in floor coordinates, and the appropriate command string assembled.

When MODE is neither zero nor one, it is set to one and defaults to position control. One frequently used subroutine in both modes is DECOMP which takes the state vector S and decomposes it to form the vectors X, V, L and q which correspond to the displacement, velocity, angular momentum and the unit quaternion vectors respectively. Throughout this module, no attempt is ever made to ensure that the magnitude of q is unity.

To ensure that SVX generates the correct command string, a series of tests were conducted using the IBM PC. First, a simple State Vector Editor is written. This editor allows one to create and edit, interactively, state vectors which are placed in sequence in a disk file. Next, a simple main program is written and linked to the SVX module. The main program consists of a driver loop that reads each state vector from the disk file and invokes SVX. The command string outputted by SVX is sent to a printer and the process is repeated until the file of state vectors is exhausted. This simple arrangement allows one to verify the correctness of SVX without disturbing it.

Since it is difficult, if not impossible, to represent the results graphically in three dimensions, state vectors are chosen such that one can easily displays the results in two dimensions. By way of example, a sequence of 60 state vectors of the form :

$$[0,0,0, 0,0,0, 0,0,0, 0,0,\sin(7.5),\cos(7.5), 1500]^T$$

is generated. This set of state vectors simulates 50 seconds of run time in which position control is used. The meaning of this state vector is that the

OMV is to remain stationary, but executes a yaw at a rate of 15° per major cycle (0.1 second). Here, we have assumed that the OMV is a disk shaped object having a uniform mass distribution and a constant mass of 1500 pounds. Note that in case of position control, the angular momentum vector is inconsequential, so a null vector is used. These figures may not be very realistic, but they are adequate for testing the SVX module. Figure 7 shows the result of a portion of the output command string. In this and subsequent figures, a circle or dot indicates the the position of the center of mass of TOM_B in floor coordinates, while an attached arrow shows its yaw. This figure depicts that TOM_B moves in a circular path and its yaw is changing at a rate of 15° per major cycle. It is noted that the radius of the circular path is equal to the distance between the centers of mass of TOM_B and the mock-up module. Thus, the mock-up module would be spinning about its Z_M axis at the same rate, exactly as expected.

When the state vectors are changed to

$$[0.5, 0, 0, 0, 0, 0, 0, 0, 0, \sin(7.5), \cos(7.5), 1500]^T$$

in position control, the path of TOM_B is shown in Figure 8. In this figure, TOM_B attempts to move in a circular path with a net displacement of 0.5 feet per major cycle. It is easy to conclude that the mock-up module would be rotating about its Z_M axis and translate along the X_M axis simultaneously, as demanded by this state vector.

CONCLUSION

Other similar tests have been conducted. For example, the state vector in the beginning of this section has been used as input for rate control, and the result is plotted in Figure 9. This and similar results have demonstrated that

the module SVX is functioning properly and that correct command strings are obtained. One must remember that the outputs of this module are commands to TOM_B, indicating the desired position, (or velocity) and attitude (or angular rates). The proper interpretation, and subsequent execution, of these commands are performed by the TOM_B Executive, and is outside the scope of the SVX module.

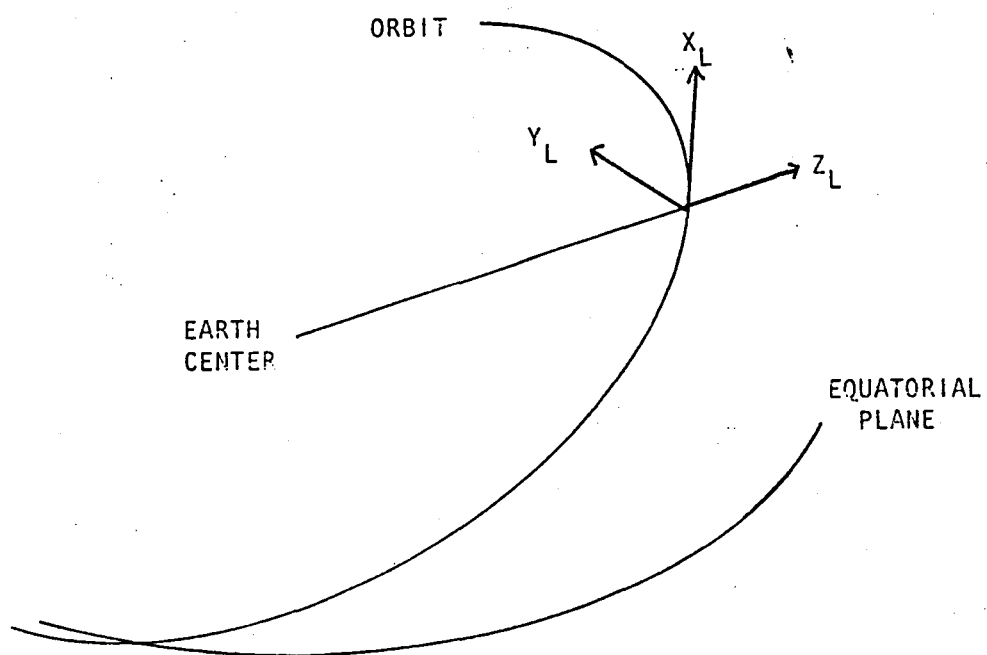


Fig. 1 Local Vertical Frame (L)

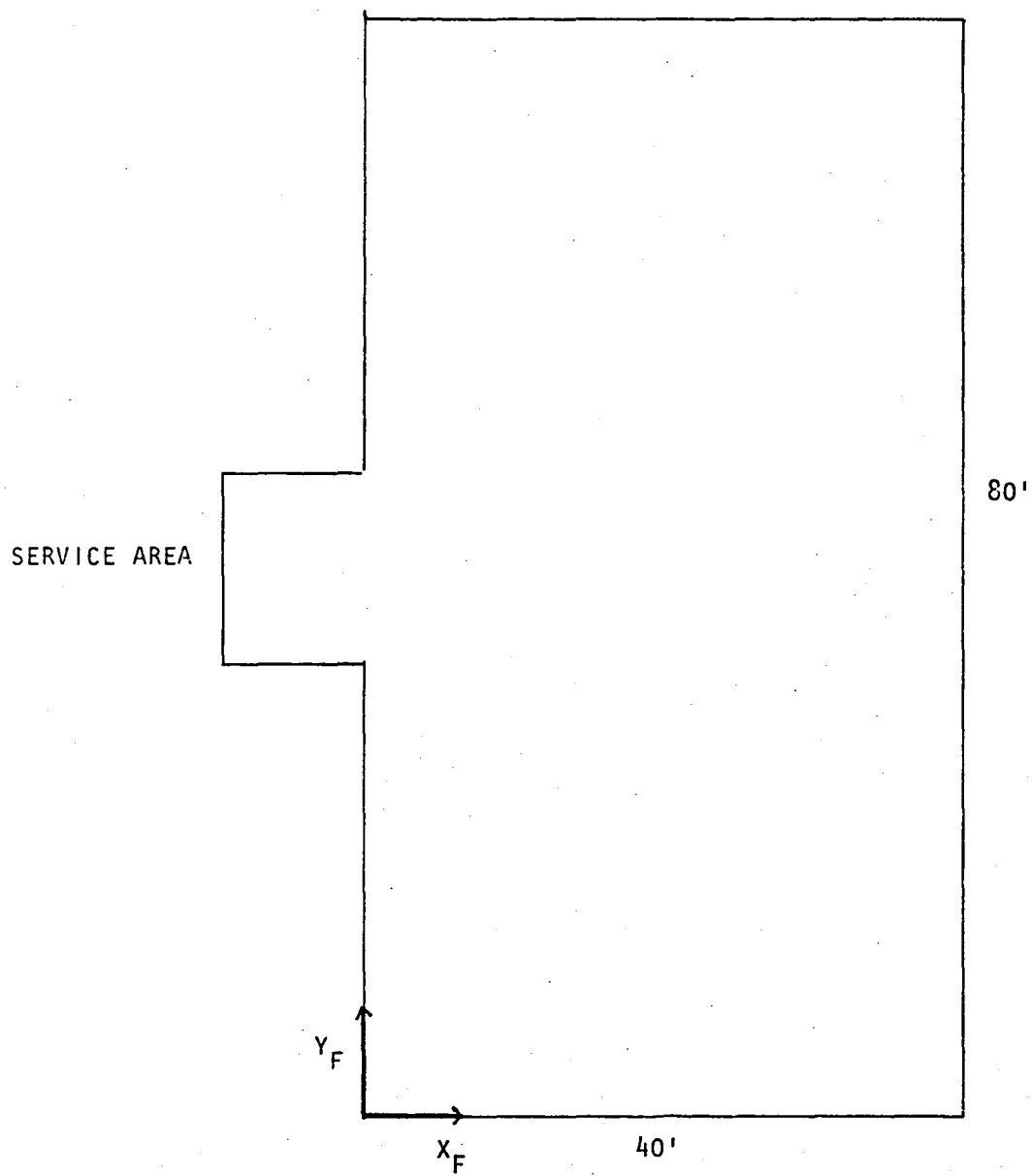


Fig 2. Floor coordinates (F)

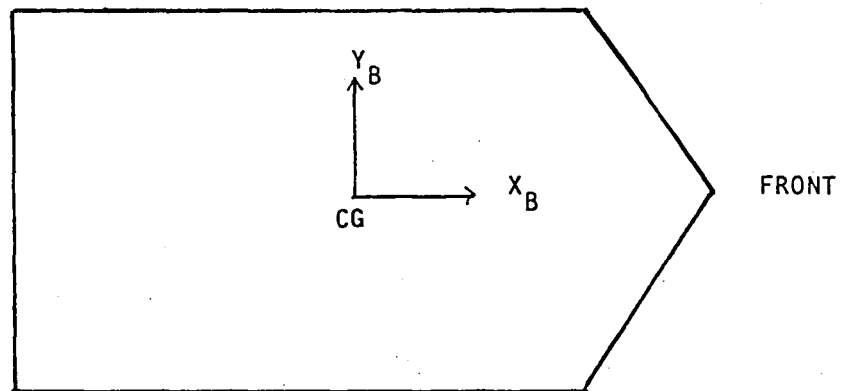


Fig 3. TOM_B Body Frame (B)

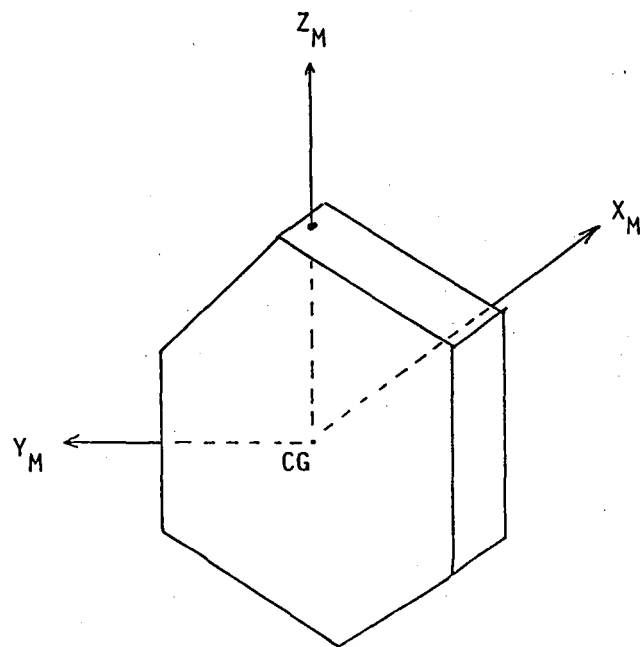


Fig 4. MOCK-UP MODULE BODY FRAME (B)

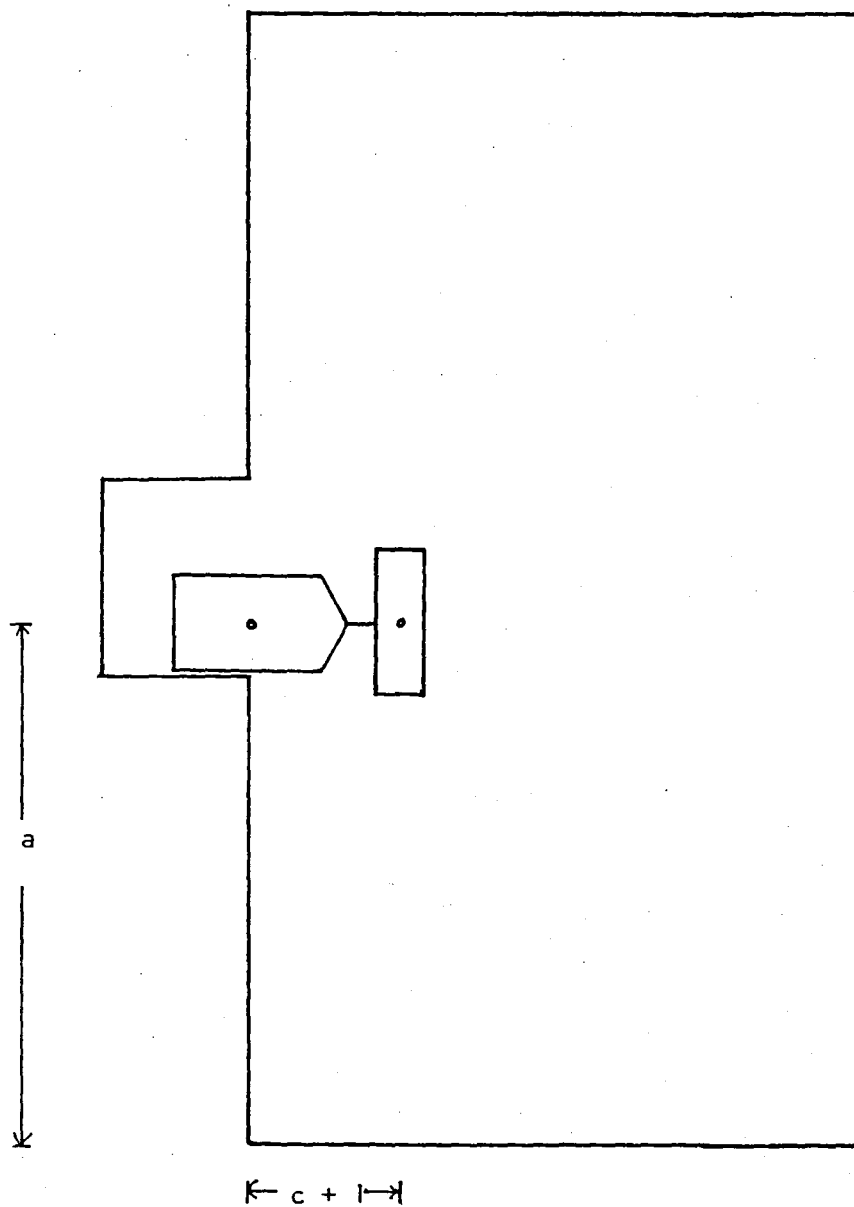


Fig 5 a) Initial position (top view)

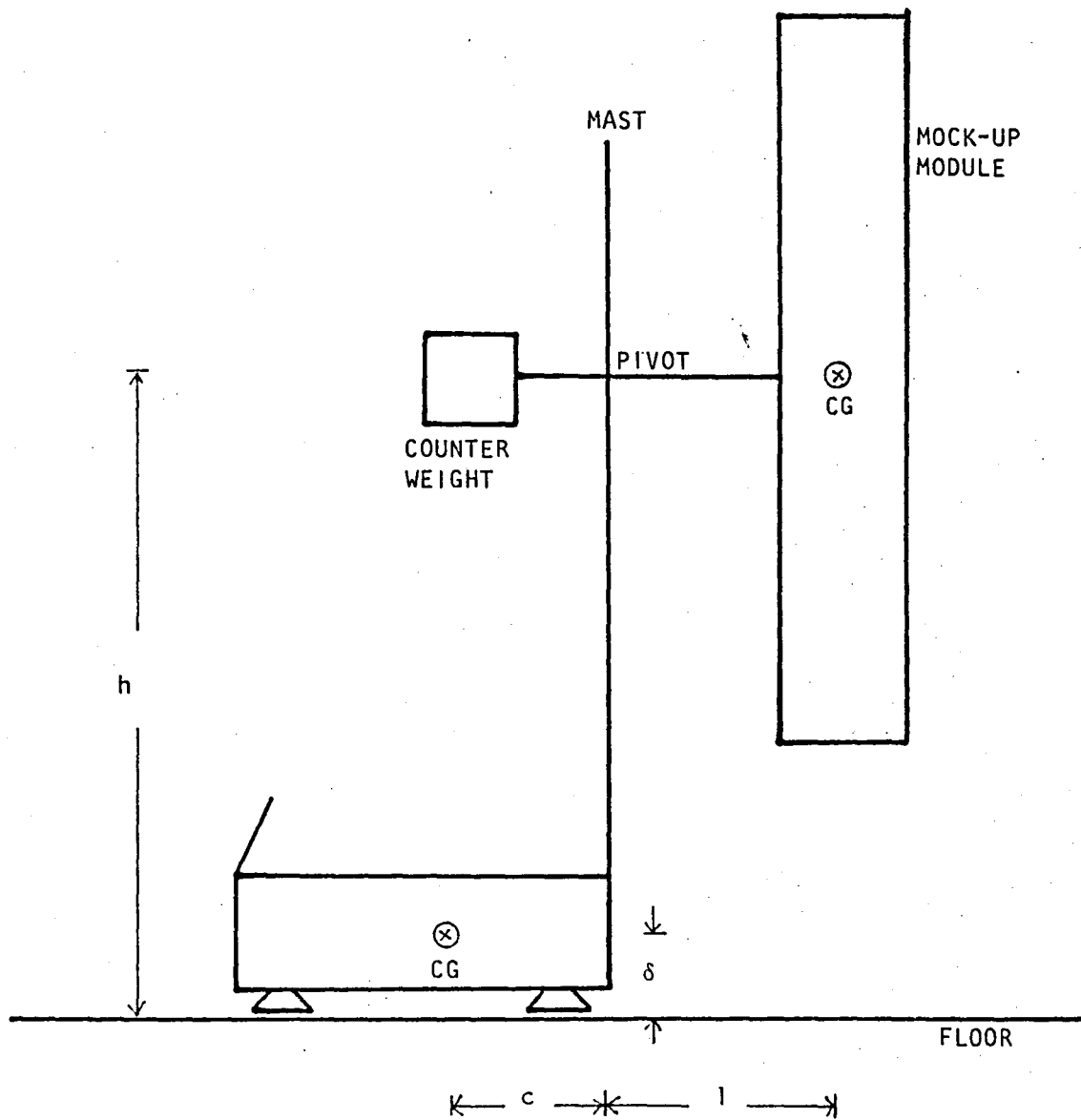


Fig 5 b) Initial position (side view)

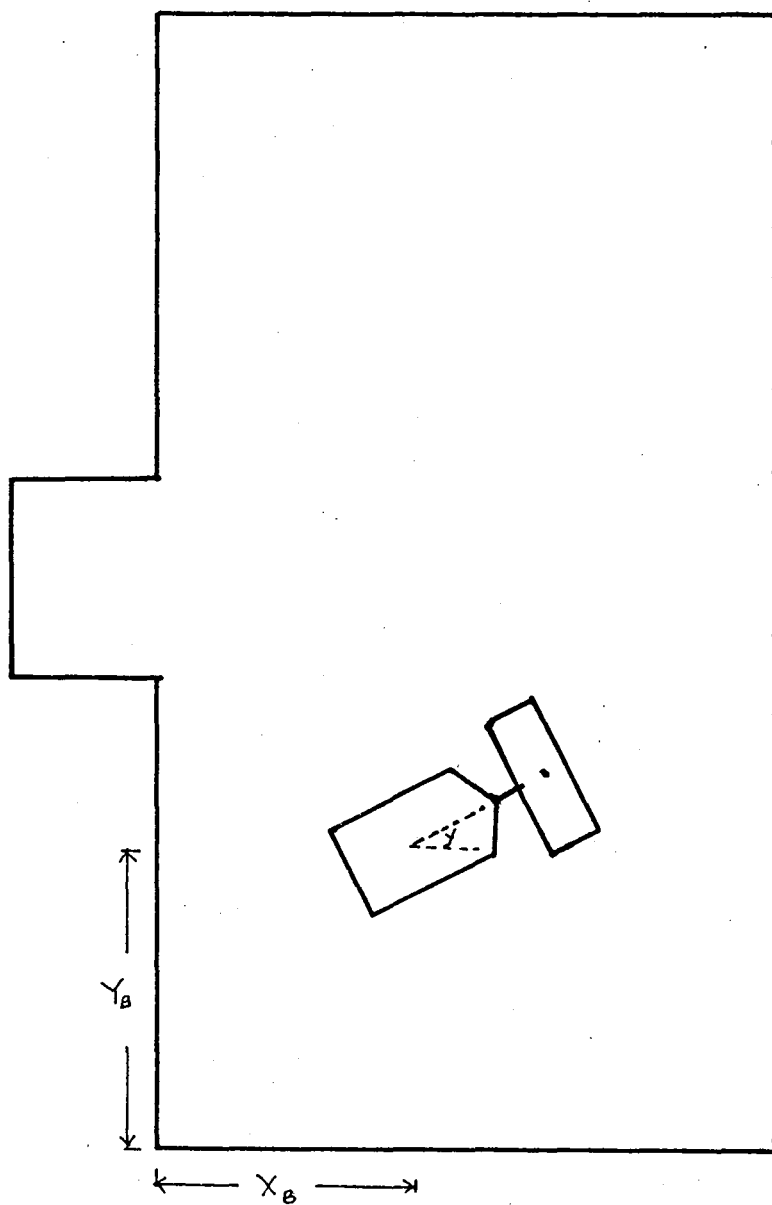


Figure 6 a) Position and yaw of TOM_B

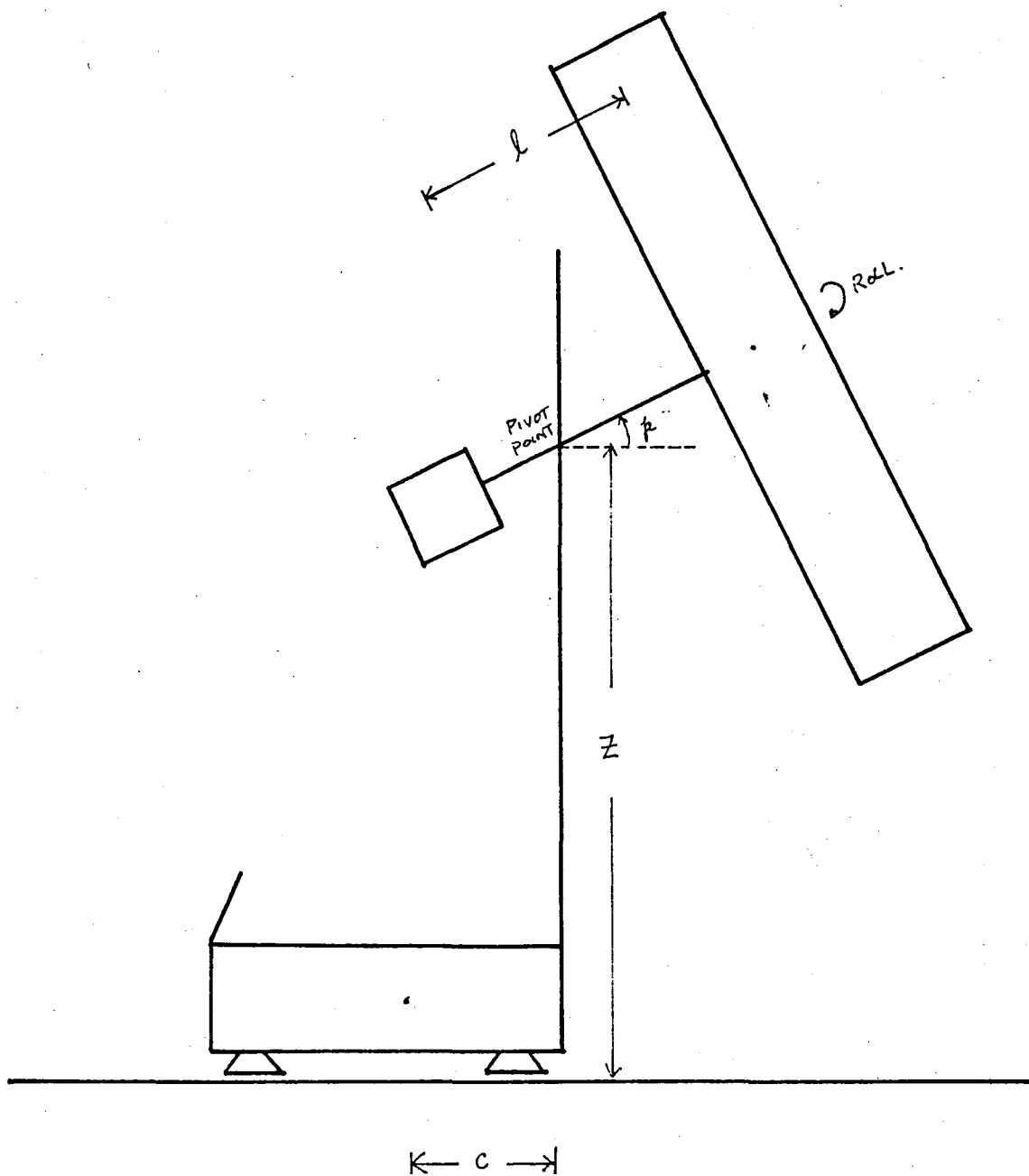


Figure 6 b) Pitch and roll of Mock-up Module

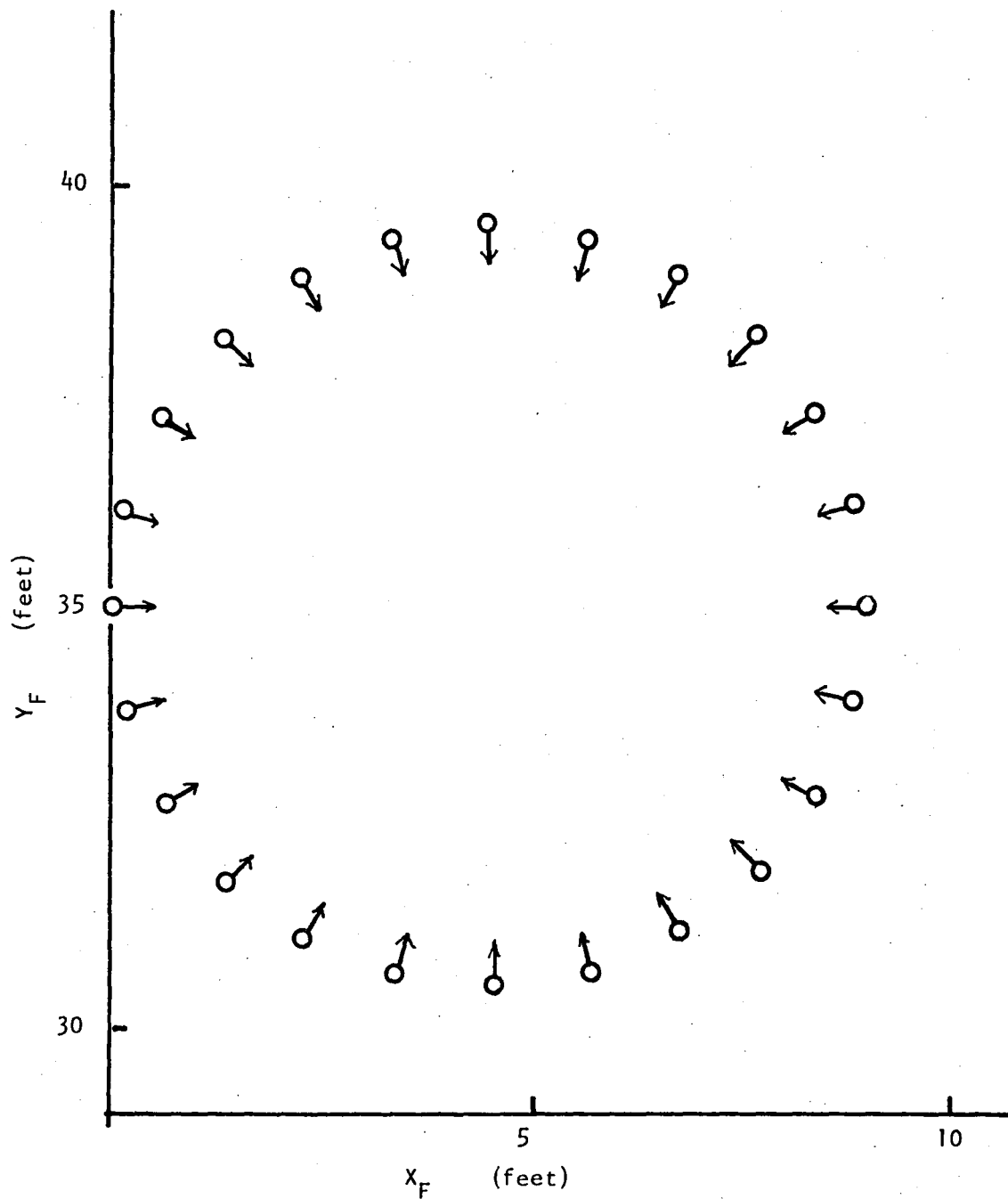


Figure 7 Position of TOM_B in floor coordinates

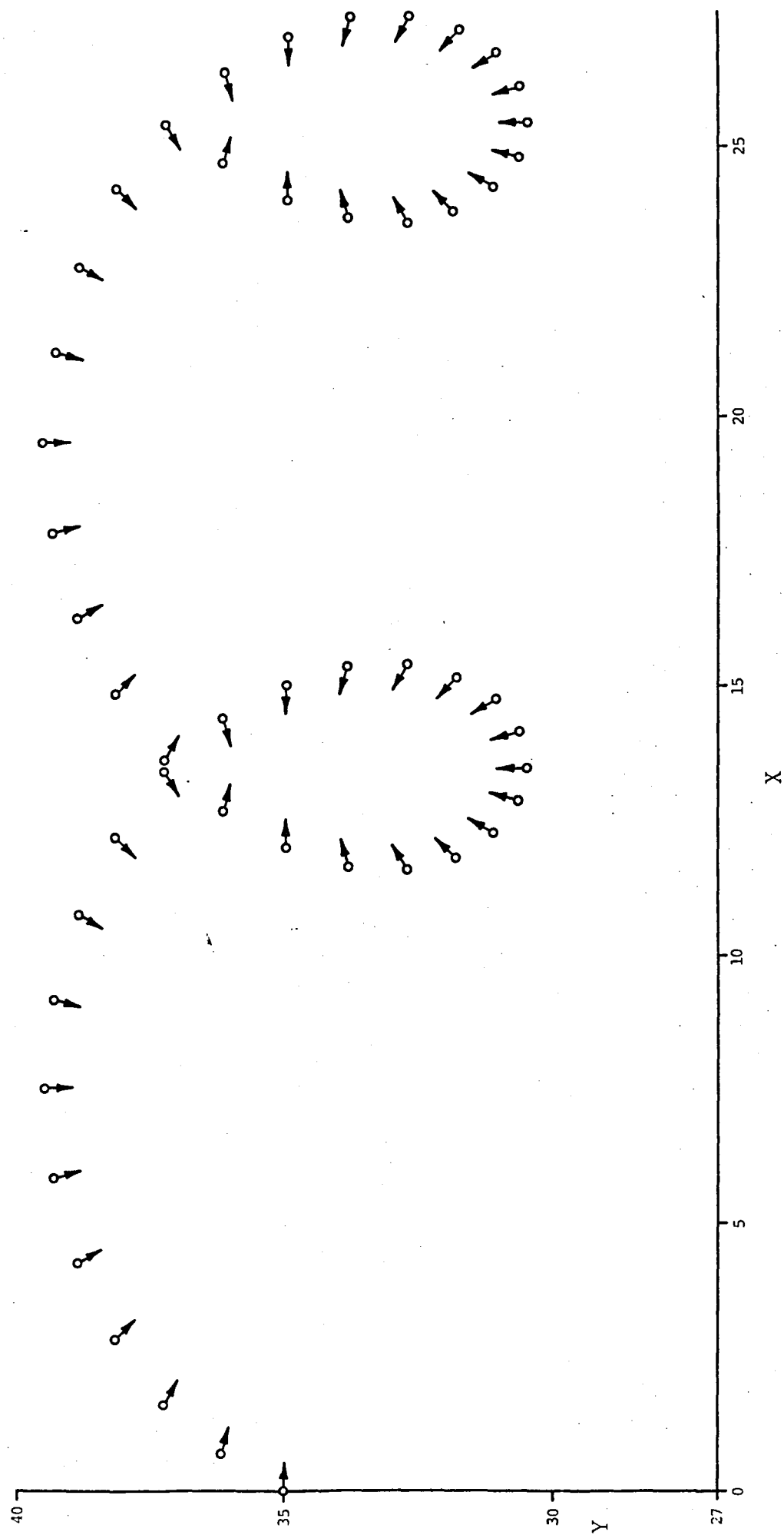


Fig. 8 Trajectory of TOM_B

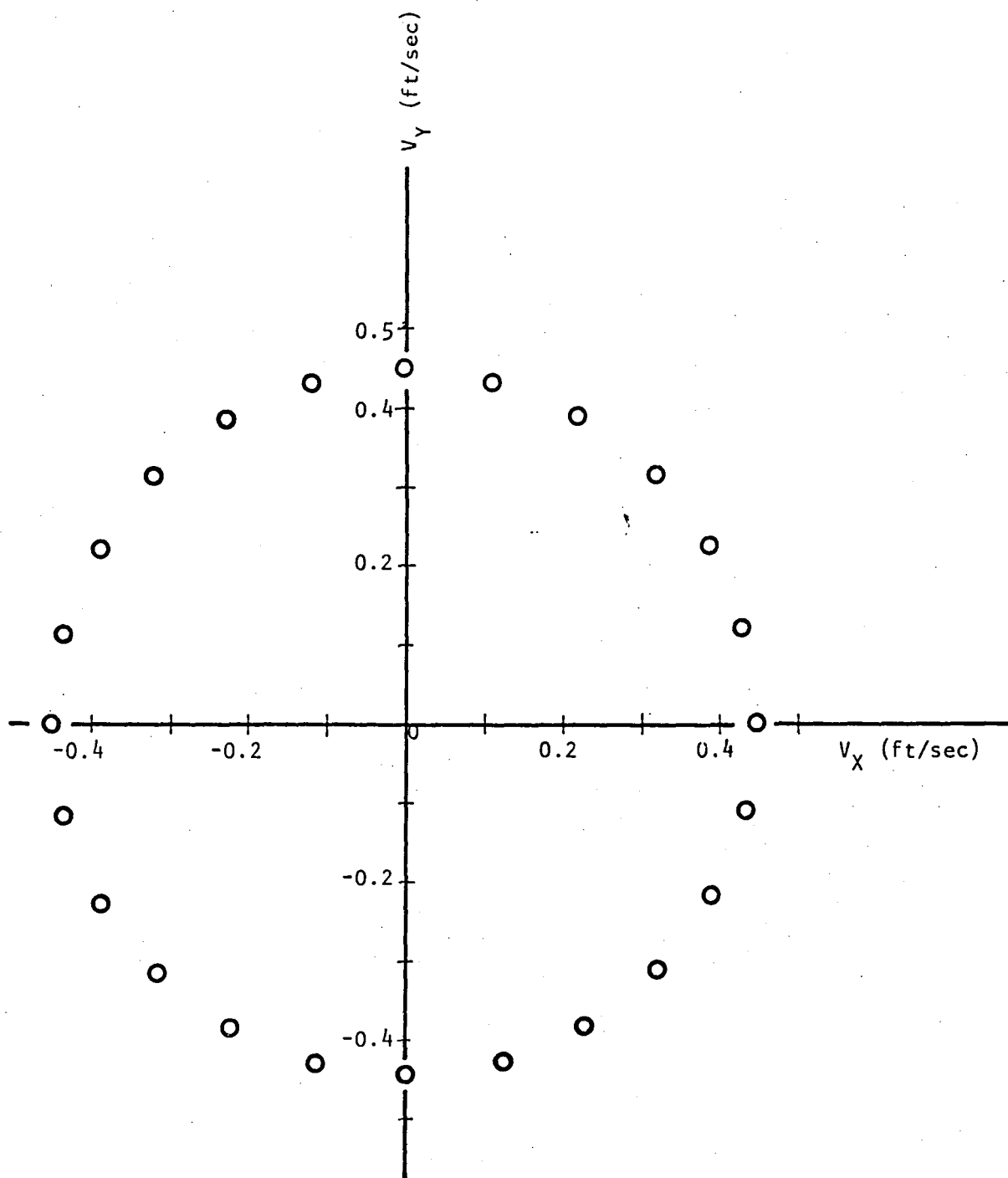


Figure 9 Velocity components of TOM_B

List of References

1. "Software Specification for Docking Simulations of the Orbital Maneuvering Vehicle OMV", J.D. Micheal, NASA internal communication. 1984.
2. "Equations of Motion for Six Degree-of-freedom TRS Simulation", J. Galaboff, Systems Dynamics Lab, 1978.
3. "Development of an Autonomous Video Rendezvous and Docking System", J.C. Tietz and T.E. Richardson, Martin Marietta Aerospace Phase 2 Final Report, 1983, NASA Contract # NAS8-34679.
4. "A New Method for Performing Digital Control System Attitude Computation Using Quaternions:", B.P. Ickes, AIAA Journal vol 8(1970)13.
5. "Design of a Terminal Pointer Hand Controller For Teleoperator Applications", E.L. Seanger & W.S. Woltosz, URS/Matrix Co. Final Report, 1973, NASA Contract # NAS8-28760.

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

1 \$PAGESIZE: 56
2 \$TITLE: '<<< S V X >>>'

3 C
4 C
5 C
6 C
7 C
8 C
9 C

STATE VECTOR TRANSFORMATION MODULE (SVX)

by

Dr. W. Teoh

U A H 1984

10 C
11 C
12 C
13 C

14 C
15 C
16 C

SUBROUTINE SVX (S, CMDRAW, MODE)

17 C
18 C
19 C
20 C

21 C
22 C
23 C
24 C
25 C
26 C

This is the state vector transformation module which accepts a 14 element state vector S of the OMV as input and generates a 6-element command string CMDRAW as output. The argument MODE conveys the following meaning :

27 C
28 C
29 C
30 C
31 C

MODE	Meaning
0	rate control
1	position control
anything else	defaults to 1

32 C
33 C

Summary of the state vector components are as follows :

34 C
35 C

Component		Meaning
1	X	position of target vehicle from the
2	Y	chase vehicle in LVF
3	Z	
4	VX	relative velocity of the two vehicles
5	VY	in LVF
6	VZ	
7	LX	angular momentum vector in LVF
8	LY	
9	LZ	
10	Q1	attitude quaternions in body frame
11	Q2	
12	Q3	
13	Q4	
14	M	instantaneous mass in kg.

36 C
37 C
38 C
39 C
40 C
41 C
42 C
43 C
44 C
45 C
46 C
47 C
48 C
49 C

Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

Summary of command string components:

component	meaning	coord system
1	YAW	body frame
2	X	floor coordinate
3	Y	floor coordinate
4	Z	floor coordinate
5	PITCH	body frame
6	ROLL	body frame
7	MODE	integer

This module maintains a local counter to process initial conditions at the start of the simulation.

```
REAL * 8    S(14)
REAL * 8    X(3), V(3), L(3), Q(4)
REAL * 8    XO(3), XM(3), E(3), XHOLD(3)
REAL * 8    IINV(3), LB(3), W(4)
REAL * 8    RPY(3), QDOT(4), QW(4,4), A(3,3)
REAL * 8    LL, UL, UA, CC, AA, HH, QQ, TX, TY, Z
REAL * 8    ROLL, PITCH, YAW, ROLDOT, PITDOT, YAWDOT
REAL * 8    Q1, Q2, SY, CY, VX, VY, VZ
```

```
INTEGER CMDRAW(7), COUNT, MODE
```

```
*** load-time initialization
```

```
DATA COUNT /0/
```

```
*** decompose state vector and process it
```

```
CALL DECOMP (S, X, V, L, Q)
```

```
IF (COUNT .NE. 0) GOTO 300
```

```
*** initialization before start
```

```
CALL ZERO (XO, 3)
```

```
*** read parameters
```

```
OPEN (1, FILE = 'SVXINT.DAT', STATUS = 'OLD')
```

```
READ (1, 20) CC, LL, AA, HH
```

```
READ (1, 20) IINV
```

<<< S V X >>>

Page 3

07-14-84

13:01:57

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

```

  99 C
 100 C      *** calculate inverse of moment of inertia tensor
 101 C
 102      DO 50 K = 1, 3
-1 103          IINV(K) = 1.0 / IINV(K)
1 104 50      CONTINUE
 105      CLOSE (1)
 106 C
 107 C      *** set conversion factors
 108 C
 109      UL = 10000.0
 110      UA = UL
 111      COUNT = COUNT + 1
 112 C
 113 C      *** set transformation matrix elements to floor coord.
 114 C
 115      E(1) = CC + LL - XO(1)
 116      E(2) = AA - XO(2)
 117      E(3) = HH - XO(3)
 118 C
 119 C      *** initialize to home orientation
 120 C
 121      CALL ZERO (RPY, 3)
 122      COUNT = COUNT + 1
 123 C
 124 300      IF (MODE .NE. 1) GO TO 400
 125 C
 126 C      *** position commands
 127 C
 128 C      *** update orientation and position
 129 C
 130      CALL QTRPY (Q, ROLL, PITCH, YAW)
 131      CALL UPDPOS (XM, X, XHOLD, E, 3)
 132 C
 133 C      *** set orientation part of the command string
 134 C
 135      CMDRAW(7) = 1
 136      CMDRAW(6) = JFIX(ROLL * UA)
 137      CMDRAW(5) = JFIX(PITCH * UA)
 138      CMDRAW(1) = JFIX(YAW * UA)
 139 C
 140 C      *** transform to TOM_B position in floor coordinates
 141 C
 142      QQ = CC + LL * DCOS(PITCH)
 143 C
 144 C      *** X-component
 145 C
 146      TX = XM(1) - QQ * DCOS(YAW)
 147      CMDRAW(2) = JFIX (TX * UL)
```

<< S V X >>>

Page 4

07-14-84

13:01:57

Microsoft FORTRAN77 V3.13 8/05/83

line# 1 7

148 C

149 C

150 C

151

152

153 C

154 C

155 C

156

157

158 C

159 C

160 C

161 C

162

163 C

164 400

165 C

166 C

167 C

168

169 C

170 C

171 C

172 C

173

174

175 C

176 C

177 C

178

179

180

181 C

182 C

183 C

184

185

186

187

188 C

189 C

190 C

191

192

193

194

195 C

196 C

*** Y-component

TY = XM(2) - QQ * DSIN(YAW)

CMDRAW(3) = JFIX (TY * UL)

*** Z-component

Z = XM(3) - LL * DSIN(PITCH)

CMDRAW(4) = JFIX (Z * UL)

*** This is a good place to call the I/O driver to

*** transmit to TOM_B, but we won't for now

RETURN

IF (MODE .NE. 0) GO TO 900

*** rate control

CALL QTRPY (Q, ROLL, PITCH, YAW)

*** form direction cosine matrix and calculate angular

*** momentum in body frame

CALL DIRCOS (A, Q)

CALL MMUL (A, L, LB, 3)

*** compute body rate

ROLDOT = IINV(1) * LB(1)

PITDOT = IINV(2) * LB(2)

YAWDOT = IINV(3) * LB(3)

*** construct orientation part of command string

CMDRAW(7) = 0

CMDRAW(6) = JFIX (ROLDOT * UA)

CMDRAW(5) = JFIX (PITDOT * UA)

CMDRAW(1) = JFIX (YAWDOT * UA)

*** compute velocity of TOM_B in floor coordinates

Q1 = LL * DSIN(PITCH) * PITDOT

Q2 = (CC + LL * DCOS(PITCH)) * YAWDOT

SY = DSIN(YAW)

CY = DCOS(YAW)

*** X-component of velocity in floor coordinate

<<< S V X >>>

Page 5
07-14-84
13:01:57

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

```

197 C
198      VX = V(1) + Q1 * CY + Q2 * SY
199      CMDRAW(2) = JFIX (VX * UL)
200 C
201 C      *** Y-component of velocity in floor coordinate
202 C
203      VY = V(2) + Q1 * SY - Q2 * CY
204      CMDRAW(3) = JFIX (VY * UL)
205 C
206 C      *** Z-component
207 C
208      VZ = V(3) - LL * DCOS(PITCH) * PITDOT
209      CMDRAW(4) = JFIX (VZ * UL)
210      RETURN
211 C
212 900      CONTINUE
213 C
214 C      *** We have an un-recognizable code, default to 1 for
215 C      *** position control
216 C
217      MODE = 1
218      GO TO 300
219 C
220 10      FORMAT (4F10.2)
221 20      FORMAT ( F15.8)
222      END

```

Name	Type	Offset	P	Class
A	REAL*8	466		
AA	REAL*8	558		
CC	REAL*8	542		
CMDRAW	INTEGER*4	4	*	
COUNT	INTEGER*4	538		
CY	REAL*8	698		
DCOS				INTRINSIC
DSIN				INTRINSIC
E	REAL*8	418		
HH	REAL*8	566		
IINV	REAL*8	442		
K	INTEGER*4	574		
L	REAL*8	370		
LB	REAL*8	394		
LL	REAL*8	550		
MODE	INTEGER*4	8	*	
PITCH	REAL*8	602		
PITDOT	REAL*8	658		
Q	REAL*8	154		
Q1	REAL*8	674		

<< S V X >>>

Page 6

07-14-84

13:01:57

Microsoft FORTRAN77 V3.13 8/05/83

line#	1	7
2	REAL*8	682
P-T	REAL*8	210
(REAL*8	618
W	REAL*8	242
QLDOT	REAL*8	650
(L	REAL*8	594
1	REAL*8	186
	REAL*8	0 *
	REAL*8	690
	REAL*8	626
Y	REAL*8	634
	REAL*8	586
	REAL*8	578
	REAL*8	98
X	REAL*8	706
	REAL*8	714
	REAL*8	722
	REAL*8	122
	REAL*8	2
	REAL*8	26
HOLD	REAL*8	74
M	REAL*8	50
N	REAL*8	610
AWDOT	REAL*8	666
	REAL*8	642

223 \$PAGE

<<< S V X >>>

Page 7

07-14-84

13:01:57

Microsoft FORTRAN77 V3.13 8/05/83

D Line# 1 7

224 C

225 C

226 SUBROUTINE DECOMP (S, X, V, L, Q)

227 C

228 C

229 C

230 C This procedure decomposes the State vector S into its components
231 C which are also vectors. They have the following meaning :

232 C

233 C	Vector	Dimension	Meaning
234 C	X	3	Position vector in LVF
235 C	V	3	Velocity vector in LVF
236 C	L	3	Angular momentum in LVF
237 C	Q	4	Unit quaternion in body frame

238 C

239 C

240 C

241 REAL * 8 S(14), X(3), V(3), L(3), Q(4)

242 C

243 CALL LD (S, X, 1, 3)

244 CALL LD (S, V, 4, 3)

245 CALL LD (S, L, 7, 3)

246 CALL LD (S, Q, 10, 4)

247 C

248 RETURN

249 END

Name	Type	Offset	P	Class
------	------	--------	---	-------

L	REAL*8	12	*	
---	--------	----	---	--

Q	REAL*8	16	*	
---	--------	----	---	--

-S	REAL*8	0	*	
----	--------	---	---	--

V	REAL*8	8	*	
---	--------	---	---	--

X	REAL*8	4	*	
---	--------	---	---	--

250 \$PAGE

<< S V X >>>

Page 8

07-14-84

13:01:57

Microsoft FORTRAN77 V3.13 8/05/83

line# 1 7

251 C

252 C

253 SUBROUTINE LD (A, B, M, N)

254 C

255 C-----

256 C

257 C This procedure copies N elements of vector A to vector B,
258 C starting at the M-th element

259 C

260 C-----

261 C

262 REAL * 8 A(14), B(N)

263 DO 100 K = 1, N

264 B(K) = A(M + K - 1)

265 100 CONTINUE

266 RETURN

267 END

ne Type Offset P Class

REAL*8 0 *

REAL*8 4 *

INTEGER*4 750

INTEGER*4 8 *

INTEGER*4 12 *

268 \$PAGE

<<< S V X >>>

Page 9

07-14-84

13:01:57

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

269 C

270 C

271 SUBROUTINE MMUL (A, B, C, N)

272 C

273 C-----

274 C

275 C This procedure performs a matrix multiplication of an NxN

276 C matrix A to an N-element column matrix B to yield an N-

277 C element column matrix C

278 C

279 C-----

280 C

281 REAL * 8 A(N,N), B(N), C(N), S

282 C

283 DO 100 I = 1, N

1 284 S = 0.0

1 285 DO 200 J = 1, N

2 286 S = S + A(I,J) * B(J)

2 287 200 CONTINUE

1 288 C(I) = S

1 289 100 CONTINUE

289 RETURN

290

291 END

Name Type Offset P Class

A REAL*8 0 *

B REAL*8 4 *

C REAL*8 8 *

I INTEGER*4 758

J INTEGER*4 774

N INTEGER*4 12 *

S REAL*8 766

292 \$PAGE

S V X >>>

Page 10

07-14-84

13:01:57

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

293 C

294 C

295 SUBROUTINE ZERO (A, N)

296 C

297 C

298 C

299 C This procedure initializes an N-element array A to zero at

300 C run time

301 C

302 C

303 C

304 REAL * 8 A(N)

305 DO 100 K = 1, N

306 A(K) = 0.0

307 100 CONTINUE

308 RETURN

309 END

ne Type Offset P Class

REAL*8 0 *

INTEGER*4 782

INTEGER*4 4 *

310 \$PAGE

<<< S V X >>>

Page 11

07-14-84

13:01:57

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

311 C

312 C

313 SUBROUTINE UPDPOS (XM, X, XHOLD, E, N)

314 C

315 C-----

316 C

317 C This procedure updates the position of the OMV in local vertical
318 C frame (XHOLD).

319 C

320 C The new position of the module in floor coordinates is then com-
321 C puted (XM)

322 C

323 C-----

324 C

325 C

326 REAL * 8 XM(N), X(N), XHOLD(N), E(N)

327 C

328 DO 100 K = 1, N

329 XHOLD(K) = X(K)

330 XM(K) = XHOLD(K) + E(K)

1 331 100 CONTINUE

332 RETURN

333 END

Name	Type	Offset	P	Class
------	------	--------	---	-------

E	REAL*8	12	*	
---	--------	----	---	--

K	INTEGER*4	790		
---	-----------	-----	--	--

	INTEGER*4	16	*	
--	-----------	----	---	--

	REAL*8	4	*	
--	--------	---	---	--

XHOLD	REAL*8	8	*	
-------	--------	---	---	--

XM	REAL*8	0	*	
----	--------	---	---	--

334 \$PAGE

S V X >>>

Page 12

07-14-84

13:01:57

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

335 C

336 C

337 INTEGER FUNCTION JFIX (RR)

338 C

339 C

340 C

341 C This procedure properly rounds a real number R to the nearest

342 C integer.

343 C

344 C

345 C

346 REAL * 8 RR

347 REAL R

348 R = RR

349 IF (R .GE. 0) THEN

350 JFIX = IFIX (R + 0.5)

351 ELSE

352 JFIX = IFIX (R - 0.5)

353 END IF

354 RETURN

355 END

me Type Offset P Class

JFIX INTRINSIC

REAL 798

REAL*8 0 *

356 \$PAGE

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

357 C

358 C

359 SUBROUTINE SETQ (QW, Q)

360 C

361 C-----

362 C

363 C This procedure constructs a 4x4 transformation matrix QW from
 364 C the attitude quaternions Q

365 C

366 C For reference, please see "Software Specifications For Docking
 367 C Simulation Of The OMV" by J. Micheals, January, 1984.

368 C

369 C-----

370 C

371 REAL * 8 QW(4,4), Q(4)

372 C

373 DO 100 I = 1, 3

1 374 DO 110 J = I+1, 4

-2 375 KK = I + J

2 376 K = KK - (KK/4) * 4

2 377 IF (K .EQ. 0) K = 2

-2 378 ISGNN = 1

2 379 IF ((J .EQ. I+1) .AND. (J.NE. 4)) ISGNN = -1

2 380 QW(I,J) = ISGNN * Q(K)

-2 381 110 CONTINUE

1 382 QW(I,I) = Q(4)

1 383 100 CONTINUE

384 QW(4,4) = Q(4)

- 385 C

386 DO 200 I = 2, 4

1 387 KK = I - 1

-1 388 DO 200 J = 1, KK

2 389 QW(I,J) = -QW(J,I)

2 390 200 CONTINUE

391 RETURN

- 392 END

Name	Type	Offset	P	Class
I	INTEGER*4	802		
ISGNN	INTEGER*4	818		
J	INTEGER*4	806		
K	INTEGER*4	814		
KK	INTEGER*4	810		
Q	REAL*8	4	*	
QW	REAL*8	0	*	

- 393 \$PAGE

S V X >>>

Page 14

07-14-84

13:01:57

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

394 C

395 C

396 SUBROUTINE DIRCOS (A, Q)

397 C

398 C

399 C

400 C This procedure takes the quaternion vector and generates

401 C a 3 X 3 direction cosine matrix A

402 C

403 C

404 C

405 C

406 REAL * 8 Q(4), A(3,3), QKS, QRS, S1

407 C

408 DO 100 K = 1, 3

409 C

410 C *** initialize diagonal elements

411 C

412 A(K,K) = Q(4) ** 2

413 DO 100 J = 1, 3

414 C

415 C *** fix up the diagonal elements

416 C

417 A(K,K) = A(K,K) + DLTKRK(K,J) * Q(J) ** 2

418 C

419 C *** now do the off-diagonal elements

420 C

421 IF (J .GT. K) THEN

422 C

423 C *** calculate index I <> J & K

424 C

425 I = 6 / (J * K)

426 C

427 C *** calculate the proper sign

428 C

429 S1 = QSIGN (K,J)

430 QKJ = Q(K) * Q(J)

431 QRS = Q(I) * Q(4) * S1

432 A(K,J) = 2.0 * (QKJ + QRS)

433 A(J,K) = 2.0 * (QKJ - QRS)

434 END IF

435 100 CONTINUE

436 RETURN

437 END

Name Type Offset P Class

A REAL*8 0 *

L INTEGER*4 838

<<< S V X >>>

Page 15

07-14-84

13:01:57

Microsoft FORTRAN77 V3.13 8/05/83

D Line# 1	7	
J	INTEGER*4	830
K	INTEGER*4	826
Q	REAL*8	4 *
QKJ	REAL	854
QKS	REAL*8	*****
QRS	REAL*8	858
S1	REAL*8	842

438 \$PAGE

< S V X >>>

Page 16

07-14-84

13:01:57

Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

439 C

440 C

441 REAL FUNCTION DLTKRK (K,J)

442 C

443 C

444 C-----

445 C

446 C

447 REAL S

448 INTEGER K, J

449 S = 1.0

450 IF (K .NE. J) S = -1.0

451 DLTKRK = S

452 RETURN

453 END

Name Type Offset P Class

INTEGER*4 4 *

INTEGER*4 0 *

REAL 866

454 \$PAGE

<<< S V X >>>

Page 17

07-14-84

13:01:57

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

455 C

456 C

457 REAL FUNCTION QSIGN(K,J)

458 C

459 C

460 C-----

461 C

462 C

463 S = 1.0

464 L = J + K

465 IF (MOD(L,2) .EQ. 0) S = -1.0

466 QSIGN = S

467 RETURN

468 END

Name	Type	Offset	P	Class
J	INTEGER*4	4	*	
K	INTEGER*4	0	*	
L	INTEGER*4	874		
MOD				INTRINSIC
S	REAL	870		

469 \$PAGE

S V X >>>

Page 18

07-14-84

13:01:57

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

470 C
471 C
472
473 C
474 C
475 C
476 C
477 C
478 C
479
480 C
481
482 C
483 C
484 C
485
486
487
488
489
490
491
492
493
494 C
495 C
496 C
497
498 C
499 C
500 C
501
502
503
504
505

SUBROUTINE QTRPY (Q, R, P, Y)

This subroutine calculates a reasonable set of roll,
pitch and yaw from the quaternion Q

REAL * 8 Q(4), R, P, Y, M, THETA, CA, CB, CG

M = DSQRT (Q(1)**2 + Q(2)**2 + Q(3)**2)

calculate direction cosines CA, CB, CG

IF (DABS(M) .LE. 1.0D-20) THEN

CA = 0.0

CB = 0.0

CG = 0.0

ELSE

CA = Q(1) / M

CB = Q(2) / M

CG = Q(3) / M

END IF

calculate angle of rotation about Euler axis

THETA = 2.0 * DACOS(Q(4))

now determine the roll, pitch and yaw

R = CA * THETA

P = CB * THETA

Y = CG * THETA

RETURN

END

Name	Type	Offset	P	Class
A	REAL*8	886		
B	REAL*8	894		
CG	REAL*8	902		
ABS				INTRINSIC
ACOS				INTRINSIC
DSQRT				INTRINSIC
	REAL*8	878		
	REAL*8	8	*	
Q	REAL*8	0	*	
R	REAL*8	4	*	

<<< S V X >>>

Page 19

07-14-84

13:01:57

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

THETA REAL*8

910

Y REAL*8

12 *

506 \$PAGE

<< S V X >>>

Page 20

07-14-84

13:01:57

Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

Label	Type	Size	Class
ECOMP			SUBROUTINE
IRCOS			SUBROUTINE
KKRK	REAL		FUNCTION
FIX	INTEGER*4		FUNCTION
D			SUBROUTINE
JL			SUBROUTINE
IGN	REAL		FUNCTION
TRPY			SUBROUTINE
TQ			SUBROUTINE
{			SUBROUTINE
PDPOS			SUBROUTINE
ERO			SUBROUTINE

Pass One No Errors Detected
506 Source Lines

Appendix 3

OMVLOT -- Source Listing

Line# 1 7

1 \$PAGESIZE: 56

2 \$TITLE: '<<< O M V P L O T >>>'

3 C

4 C-----

5 C

6 C

7 C

Program : O M V P L O T

8 C

9 C

10 C

by

11 C

12 C

13 C

Dr. W. Teoh

14 C

15 C

16 C

U A H 1984

17 C

18 C

19 C

20 C

21 C

22 C-----

23 C

24 C

25 C

This is a graphical package that accepts a command string and uses this information to generate and display the position and orientation of TOM_B and the attached mock-up module in two dimensions. One can choose to display either the top or side view of the system.

26 C

27 C

28 C

29 C

30 C

31 C

This package is developed in FORTRAN 77 to run on an IBM PC with at least 128K of RAM, and fitted with a TECMAR GRAPHICS MASTER board. An IBM Monochrome monitor is used for the actual display. The resolution in this work is chosen to be 640 x 350.

32 C

33 C

34 C

35 C

36 C

37 C

38 C-----

39 C

40 C

41 C

42 C

43

SUBROUTINE SIDEVIEW (H, X, P)

44 C

45 C

46 C-----

47 C

This procedure produces a side view of TOM_B and the attached mock-up module. The perspective is always in the direction of +1 axis of the body fixed coordinate

48 C

49 C

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

```
50 C      system
51 C
52 C
53 C-----
54 C
55 C
56      REAL * 8      H, X, P, C, S
57      REAL          XFORM(3,3), SDFORM(3,3), VO(3,10), V(3,10)
58      REAL          ROT(3,3), FLOOR(3,3), V1(3,10)
59      REAL          CC, DD, LL, RR, WW, TT
60      INTEGER       FLAG, N, CLR, EF,EEF, PRTFG
61 C
62      COMMON /MG/ FLAG, CC, DD, LL, RR, WW, TT
63      COMMON /MF/ XFORM, SDFORM, VO, V1
64      COMMON /ME/ EF, EEF, PRTFG
65 C
66 C
67      N = 10
68      AA = 1.0
69 C
70 C      ***  define mock-up module shape at origin
71 C
72      DO 100 K = 1, N
73          V(3, K) = 1.0
-1 74 100      CONTINUE
75 C                                  << point A >>
76      V(1,1) = TT
77      V(2,1) = -DD
- 78 C                                  << point B >>
79      V(1,2) = -TT
80      V(2,2) = -DD
- 81 C                                  << point C >>
82      V(1,3) = -TT
83      V(2,3) = DD
- 84 C                                  << point D >>
85      V(1,4) = TT
86      V(2,4) = DD
87 C
88 C      ***  rotate it by P radians
89 C
90      CALL SINCOS (P, S, C)
91      CALL NOTHING (ROT, 3)
92      ROT(1,1) = C
93      ROT(1,2) = -S
94      ROT(2,1) = S
95      ROT(2,2) = C
96      CALL XMUL (ROT, V, 4)
97 C
98 C      ***  calculate translation
```

< O M V P L O T >>>

Page 3
07-14-84
14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

```
Line# 1 7
99 C
100 PX = CC + LL * C
101 PY = H + LL * S
102 C
103 C *** move the rotated module out there
104 C
105 CALL NOTING (ROT, 3)
106 ROT(1,3) = PX
107 ROT(2,3) = PY
108 CALL XMUL (ROT, V, 4)
109 C
110 C *** now calculate the shape of the base
111 C
112 C XX = X + CC
113 C << point E >>
114 V(1,5) = CC
115 V(2,5) = H
116 C << point F >>
117 V(1,6) = CC
118 V(2,6) = AA
119 C << point G >>
120 V(1,7) = CC
121 V(2,7) = 0.
122 C << point H >>
123 V(1,8) = -RR
124 V(2,8) = 0.
125 C << point I >>
126 V(1,9) = -RR
127 V(2,9) = AA
128 C
129 V(1,10) = PX
130 V(2,10) = PY
131 C
132 C *** Transform to floor coordinates
133 C
134 CALL NOTING (FLOOR, 3)
135 FLOOR(1,3) = X
136 CALL XMUL (FLOOR, V, N)
137 C
138 C *** transform to screen coordinates
139 C
140 CALL XMUL (SDFORM, V, N)
141 C
142 C *** erase old picture
143 C
144 CALL DRWFLR (VO)
145 IF ((EF .EQ. 0) .AND. (EEF .NE. 0)) THEN
146 CLR = 0
147 CALL SDRAW (V1, N, CLR)
```

<<< O M V P L O T >>>

Page 4
07-14-84
14:03:20

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

```
148      END IF
149      CLR = 1
150      CALL SDRAW (V, N, CLR)
151      CALL MOVE (V, V1, N)
152      EEf = 1
153 C
154      RETURN
155      END
```

Name	Type	Offset	P	Class
------	------	--------	---	-------

-AA	REAL	198		
C	REAL*8	218		
CC	REAL	4	/MG	/
-CLR	INTEGER*4	234		
DD	REAL	8	/MG	/
EEF	INTEGER*4	4	/ME	/
EF	INTEGER*4	0	/ME	/
-FLAG	INTEGER*4	0	/MG	/
FLOOR	REAL	158		
H	REAL*8	0 *		
-K	INTEGER*4	202		
LL	REAL	12	/MG	/
N	INTEGER*4	194		
-P	REAL*8	8 *		
PRTFG	INTEGER*4	8	/ME	/
PX	REAL	226		
PY	REAL	230		
-ROT	REAL	122		
RR	REAL	16	/MG	/
S	REAL*8	210		
-SDFORM	REAL	36	/MF	/
IT	REAL	24	/MG	/
V	REAL	2		
-VO	REAL	72	/MF	/
V1	REAL	192	/MF	/
-W	REAL	20	/MG	/
X	REAL*8	4 *		
-YFORM	REAL	0	/MF	/

156 \$PAGE

line# 1 7

157 C

158 C

159 SUBROUTINE SDRAW (V, N, CLR)

160 C

161 C

162 C-----

163 C

164 C

165 C This procedure draws the side view of TOM_B

166 C

167 C

168 C-----

169 C

170 C

171 REAL V(3,10)

172 INTEGER N, CLR, X1, X2, Y1, Y2

173 C

174 C *** draw mobile base

175 C

176 CALL RCT (V, 5, CLR)

177 C

178 C *** draw linkage

179 C

180 X1 = V(1,6)

181 Y1 = V(2,6)

182 X2 = V(1,5)

183 Y2 = V(2,5)

184 CALL LINE (X1, Y1, X2, Y2, CLR)

185 X1 = V(1,10)

186 Y1 = V(2,10)

187 CALL LINE (X2, Y2, X1, Y1, CLR)

188 C

189 C *** draw mock-up module

190 C

191 CALL RCT (V, 0, CLR)

192 CALL PURGE

193 CALL GRFRDY

194 C

195 CALL HOME

196 C

197 RETURN

198 END

ame	Type	Offset	P	Class
TR	INTEGER*4	8	*	
	INTEGER*4	4	*	
	REAL	0	*	
1	INTEGER*4	238		

<<< O M V P L O T >>>

Page 6

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

D Line# 1 7

X2 INTEGER*4 246

Y1 INTEGER*4 242

Y2 INTEGER*4 250

199 \$PAGE

< OMV PLOT >>>

Page 7

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

200 C

201 C

202 SUBROUTINE RCT (V, OFF, CLR)

203 C

204 C

205 C

206 C

207 C This procedure draws a rectangle

208 C

209 C

210 C

211 C

212 C

213 REAL V(3,10)

214 INTEGER OFF, CLR, X(10), Y(10)

215 C

216 DO 100 K = 1, 4

217 J = K + OFF

218 X(K) = V(1,J)

219 Y(K) = V(2,J)

220 100 CONTINUE

221 CALL POLYGN(4, X, Y, CLR)

222 RETURN

223 END

me Type Offset P Class

LR INTEGER*4 8 *

INTEGER*4 338

INTEGER*4 334

OFF INTEGER*4 4 *

REAL 0 *

INTEGER*4 254

INTEGER*4 294

224 \$PAGE

D Line# 1 7

225 SUBROUTINE PLOT (CMD)

226 C

227 C

228 C-----

229 C

230 C

231 C

232 C

233 C

234 C

235 C

236 C

237 C-----

238 C

239 C

240 INTEGER CMD(7), FLAG

241 REAL * 8 X, Y, T, UL, UA, H

242 REAL XFORM(3,3),SDFORM(3,3),CC,LL,DD,RR,WW,TT

243 REAL VO(3,10), V1(3,10)

244 C

245 COMMON /MG/ FLAG, CC, DD, LL, RR, WW, TT

246 COMMON /MF/ XFORM, SDFORM, VO, V1

247 C

248 UL = 10000.0

249 UA = UL

250 C

251 IF (FLAG .EQ. 0) THEN

252 T = CMD(1) / UA

253 X = CMD(2) / UL

254 Y = CMD(3) / UL

255 CALL TOPVIEW (X, Y, T)

256 ELSE

257 H = CMD(4) / UL

258 X = CMD(2) / UL

259 T = CMD(5) / UA

260 CALL SIDEVIEW (H, X, T)

261 END IF

262 C

263 RETURN

264 END

Name	Type	Offset	P	Class
CC	REAL	4	/MG	/
CMD	INTEGER*4	0	*	
D	REAL	8	/MG	/
FLAG	INTEGER*4	0	/MG	/
H	REAL*8	382		
L	REAL	12	/MG	/

OMV PLOT >>>

Page 9

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

line# 1	7			
REAL	16	/MG	/	
FORM REAL	36	/MF	/	
REAL*8	358			
REAL	24	/MG	/	
REAL*8	350			
REAL*8	342			
REAL	72	/MF	/	
REAL	192	/MF	/	
REAL	20	/MG	/	
REAL*8	366			
FORM REAL	0	/MF	/	
REAL*8	374			

265 \$PAGE

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

266 C

267 C

268 SUBROUTINE TOPVIEW (PX, PY, THETA)

269 C

270 C-----

271 C

272 C

273 C This procedure constructs the top view of TOM_B. No

274 C correction to perspective distortion is implemented

275 C

276 C

277 C-----

278 C

279 REAL * 8 PX, PY, THETA, S, C

280 REAL V(3,10), VO(3,10), SDFORM(3,3)

281 REAL ROT(3,3), FLOOR(3,3), XFORM(3,3)

282 REAL CC, DD, LL, RR, WW, TT, V1(3,10)

283 INTEGER FLAG, N, CLR, EF, EEF, PRTFG

284 C

285 COMMON /MG/ FLAG, CC, DD, LL, RR, WW, TT

286 COMMON /MF/ XFORM, SDFORM, VO, V1

287 COMMON /ME/ EF, EEF, PRTFG

288 C

289 N = 10

290 C

291 C *** get TOM_B shape at the origin

292 C

293 CALL ORGPOS (V, N)

294 C

295 C *** rotate by THETA if needed

296 C

297 C IF (THETA .NE. 0.0) THEN

298 C *** construct rotation matrix

299 CALL NOTNG (ROT, 3)

300 CALL SINCOS (THETA, S, C)

301 ROT(1,1) = C

302 ROT(1,2) = -S

303 ROT(2,1) = S

304 ROT(2,2) = C

305 C *** rotate it

306 CALL XMUL (ROT, V, N)

307 C END IF

308 C

309 C *** transform to floor coordinates

310 C

311 CALL NOTNG (FLOOR, 3)

312 FLOOR(1,3) = PX

313 FLOOR(2,3) = PY

314 CALL XMUL (FLOOR, V, N)

Line# 1 7 Microsoft FORTRAN77 V3.13 8/05/83

```
315 C
316 C      *** transform to screen coordinates
317 C
318 C      CALL XMUL (XFORM, V, N)
319 C
320 C      *** get ready to draw, but first erase old picture
321 C
322 C      CALL DRWFLR (V1)
323 C      IF ((EF .EQ. 0) .AND. (EEF .NE. 0)) THEN
324 C          CLR = 0
325 C          CALL DRAW (VO, N, CLR)
326 C      END IF
327 C
328 C      CLR = 1
329 C      CALL DRAW (V, N, CLR)
330 C      CALL MOVE (V, VO, N)
331 C      EEF = 1
332 C
333 C      RETURN
334 C      END
```

Line	Type	Offset	P	Class
	REAL*8	594		
2	REAL	4	/MG	/
3	INTEGER*4	602		
4	REAL	8	/MG	/
EF	INTEGER*4	4	/ME	/
5	INTEGER*4	0	/ME	/
AG	INTEGER*4	0	/MG	/
DOOR	REAL	546		
6	REAL	12	/MG	/
	INTEGER*4	582		
RTFG	INTEGER*4	8	/ME	/
7	REAL*8	0 *		
	REAL*8	4 *		
8	REAL	510		
9	REAL	16	/MG	/
	REAL*8	586		
FORM	REAL	36	/MF	/
IETA	REAL*8	8 *		
10	REAL	24	/MG	/
	REAL	390		
11	REAL	72	/MF	/
12	REAL	192	/MF	/
	REAL	20	/MG	/
ORM	REAL	0	/MF	/
335	\$PAGE			

<<< O M V P L O T >>>

Page 12

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

D Line# 1 7

336 C

337 C

338 SUBROUTINE MOVE (V, VO, N)

339 C

340 C

341 C

342 C

343 C

344 C This procedure saves the shape vector V

345 C

346 C

347 C

348 C

349 C

350 REAL V(3,10), VO(3,10)

351 C

352 DO 100 K = 1, N

353 DO 100 J = 1, 3

354 VO(J,K) = V(J,K)

355 100 CONTINUE

356 C

357 RETURN

358 END

Name	Type	Offset	P	Class
------	------	--------	---	-------

J	INTEGER*4	614		
---	-----------	-----	--	--

K	INTEGER*4	606		
---	-----------	-----	--	--

N	INTEGER*4	8	*	
---	-----------	---	---	--

V	REAL	0	*	
---	------	---	---	--

VO	REAL	4	*	
----	------	---	---	--

359 \$PAGE

<< O M V P L O T >>>

Page 13

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

360 C

361 C

362 SUBROUTINE NOTHING (A, N)

363 C

364 C

365 C

366 C

367 C

368 C This procedure initializes an N x N matrix A to a

369 C unit matrix

370 C

371 C

372 C

373 C

374 C

375 REAL A(N,N)

376 C

377 DO 100 K = 1, N

378 DO 200 J = 1, N

379 A(K,J) = 0.0

380 200 CONTINUE

381 A(K,K) = 1.0

382 100 CONTINUE

383 C

384 RETURN

385 END

ame Type Offset P Class

REAL 0 *

INTEGER*4 626

INTEGER*4 618

INTEGER*4 4 *

386 \$PAGE

D Line# 1 7

387 C

388 C

389

SUBROUTINE XMUL (R, V, N)

390 C

391 C

392 C

393 C

394 C

395 C

This procedure uses a transformation matrix R and
transforms the shape vector V having N columns

396 C

397 C

398 C

399 C

400 C

401 C

402

REAL R(3,3), V(3,10), T(3), S

403

INTEGER ROW, COL

404 C

405

DO 100 COL = 1, N

406

DO 200 ROW = 1, 3

407

S = 0.0

408

DO 300 J = 1, 3

409

S = S + R(ROW,J) * V(J, COL)

410 300

CONTINUE

411

T(ROW) = S

412 200

CONTINUE

413

DO 400 L = 1, 3

414

V(L,COL) = T(L)

415 400

CONTINUE

416 100

CONTINUE

417 C

418

RETURN

419

END

Name	Type	Offset	P	Class
COL	INTEGER*4	646		
J	INTEGER*4	662		
L	INTEGER*4	666		
N	INTEGER*4	8	*	
R	REAL	0	*	
ROW	INTEGER*4	654		
S	REAL	658		
T	REAL	634		
V	REAL	4	*	

420 \$PAGE

Line# 1 7

421 C

422 C

423 SUBROUTINE ORGPOS (V, N)

424 C

425 C

426 C

427 C

428 C

429 C This procedure calculates the shape vector V of TOM_B
430 C at the origin. Only the top view is considered here.

431 C

432 C

433 C

434 C

435 C

436 REAL V(3,10), XFORM(3,3), VO(3,10), W(2)

437 REAL V1(3,10)

438 REAL CC, DD, LL, RR, WW, CL, SDFORM(3,3)

439 INTEGER FLAG, CORNR(2,2), EF, EEF, PRTFG

440 C

441 COMMON /MG/ FLAG, CC, DD, LL, RR, WW, TT

442 COMMON /MF/ XFORM, SDFORM, VO, V1

443 COMMON /ME/ EF, EEF, PRTFG

444 C

445 C

446 DO 100 K = 1, N

447 V(3, K) = 1.0

448 100 CONTINUE

449 C

450 C *** set up shape matrix V

451 C

452 CL = CC + LL

453 C Corner << A >>

454 V(1, 1) = CC

455 V(2, 1) = 0

456 C Corner << B >>

457 V(1, 2) = CC

458 V(2, 2) = -WW

459 C Corner << C >>

460 V(1, 3) = -RR

461 V(2, 3) = -WW

462 C Corner << D >>

463 V(1, 4) = -RR

464 V(2, 4) = WW

465 C Corner << E >>

466 V(1, 5) = CC

467 V(2, 5) = WW

468 C Corner << MM >>

469 V(1, 6) = CL

<<< O M V P L O T >>>

Page 16
07-14-84
14:03:20

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

```

470      V(2, 6) = 0
471 C
472      V(1, 7) = CL + TT
473      V(2, 7) = -DD
474 C
475      V(1, 8) = CL - TT
476      V(2, 8) = -DD
477 C
478      V(1, 9) = CL - TT
479      V(2, 9) = DD
480 C
481      V(1,10) = CL + TT
482      V(2,10) = DD
483 C
484      RETURN
485      END

```

Corner << F >>

Corner << G >>

Corner << H >>

Corner << I >>

Name	Type	Offset	P	Class
CC	REAL	4	/MG	/
CL	REAL	702		
-CORNR	INTEGER*4	678		
DD	REAL	8	/MG	/
EEF	INTEGER*4	4	/ME	/
_EF	INTEGER*4	0	/ME	/
FLAG	INTEGER*4	0	/MG	/
K	INTEGER*4	694		
LL	REAL	12	/MG	/
-V	INTEGER*4	4 *		
PRTFG	INTEGER*4	8	/ME	/
RR	REAL	16	/MG	/
-SDFORM	REAL	36	/MF	/
TT	REAL	24	/MG	/
V	REAL	0 *		
VO	REAL	72	/MF	/
V1	REAL	192	/MF	/
W	REAL	670		
WW	REAL	20	/MG	/
-FORM	REAL	0	/MF	/

486 \$PAGE

Line# 1 7

487 C

488 C

489 SUBROUTINE INITPL

490 C

491 C

492 C

493 C

494 C

495 C

496 C

497 C

498 C

499 C

500 C

501 C

502 C

503

REAL VO(3,10),XFORM(3,3),SDFORM(3,3), W(2)

504

REAL CC, DD, LL, RR, WW, TT, V1(3,10)

505 C

REAL CORNR(2,2), W(2)

506

INTEGER FLAG, EF, CORNR(2,2), EEF, CORNS(2,2), PRTFG

507 C

508

COMMON /MG/ FLAG, CC, DD, LL, RR, WW, TT

509

COMMON /MF/ XFORM, SDFORM, VO, V1

510

COMMON /ME/ EF, EEF, PRTFG

511 C

512

EEF = 0

513

OPEN (7, FILE = 'SIZE.DAT')

514

READ (7, 10) CC, DD, LL, RR, WW, TT

515

DO 200 K = 1, 2

516

READ (7, 20) (CORN(R,K,J), J=1, 2)

517 200

CONTINUE

518

W(1) = 12.2

519

W(2) = 24.4

520

CALL CORDX (CORN(R, XFORM, W)

521 C

522

DO 300 K = 1, 2

523

READ (7, 20) (CORNS(K,J), J=1,2)

524 300

CONTINUE

525

W(1) = 12.2

526

W(2) = 6.096

527

CALL CORDX (CORNS, SDFORM, W)

528 C

529

READ (7,20) EF

530

READ (7, 20) FLAG

531

READ (7, 20) PRTFG

532

CLOSE (7)

533

FLG = 1

534 C

535 C

*** calculate floor shape


```
Line# 1      7
536 C
537          JW = 30
538          JL = 44
539 C
540          IF (FLAG .EQ. 0) THEN
541              J1      = CORNR(1,1)
542              L1      = CORNR(1,2)
543              J2      = CORNR(2,1)
544              L2      = CORNR(2,2)
545              JJ      = (L2 - L1 + 1) / 2
546              V1(1,1) = J1
547              V1(2,1) = L1
548              V1(1,2) = J2
549              V1(2,2) = L1
550              V1(1,3) = J2
551              V1(2,3) = L2
552              V1(1,4) = J1
553              V1(2,4) = L2
554              V1(1,5) = J1
555              V1(2,5) = L2 + JW - JJ
556              V1(1,6) = J1 - JL
557              V1(2,6) = L2 + JW - JJ
558              V1(1,7) = J1 - JL
559              V1(2,7) = L2 - JL - JJ
560              V1(1,8) = J1
561              V1(2,8) = L2 - JL - JJ
562              V1(1,9) = -1000.0
563              V1(2,9) = -1000.0
564          ELSE
565              J1      = CORNS(1,1)
566              L1      = CORNS(1,2)
567              J2      = CORNS(2,1)
568              L2      = CORNS(2,2)
569              VO(1,1) = J1 - JL
570              VO(2,1) = L2 + 1
571              VO(1,2) = J2 + JL
572              VO(2,2) = L2 + 1
573              VO(1,3) = -1000.0
574              VO(2,3) = -1000.0
575          END IF
576 C
577          CALL GRAFICS
578          RETURN
579 10      FORMAT (F15.8)
580 20      FORMAT (I3)
581      END
```

ame	Type	Offset	P	Class
-----	------	--------	---	-------

<<< O M V P L O T >>>

Page 19

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1	7			
CC	REAL	4	/MG	/
ORNR	INTEGER*4	714		
ORNS	INTEGER*4	730		
DD	REAL	8	/MG	/
PEF	INTEGER*4	4	/ME	/
F	INTEGER*4	0	/ME	/
FLAG	INTEGER*4	0	/MG	/
ELG	REAL	758		
	INTEGER*4	750		
J1	INTEGER*4	770		
J2	INTEGER*4	778		
J	INTEGER*4	786		
L	INTEGER*4	766		
JW	INTEGER*4	762		
K	INTEGER*4	746		
1	INTEGER*4	774		
L2	INTEGER*4	782		
LL	REAL	12	/MG	/
RTFG	INTEGER*4	8	/ME	/
RR	REAL	16	/MG	/
SDFORM	REAL	36	/MF	/
T	REAL	24	/MG	/
O	REAL	72	/MF	/
V1	REAL	192	/MF	/
E	REAL	706		
N	REAL	20	/MG	/
KFORM	REAL	0	/MF	/

582 \$PAGE

<< O M V P L O T >>>

Page 20

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

583 C

584 C

585

586 C

587 C

588 C

589 C

590 C

591 C

592 C

593 C

594 C

595 C

596

597

598 C

599

600 C

601 C

602 100

603

604

605

606

607 C

608 C

609

610

611

SUBROUTINE DRWFLR (V)

This subroutine draws the floor portion of graphics

REAL V(3,10)
INTEGER CT, X(10), Y(10)

CT = 1

REPEAT

K = CT

X(K) = V(1,K)

Y(K) = V(2,K)

CT = CT + 1

IF (V(1,CT) .GE. -100.0) GO TO 100

UNTIL V(1,CT) < -100.0

CALL POLYGN (K, X, Y, 1)

RETURN

END

ame Type Offset P Class

INTEGER*4 884

INTEGER*4 888

REAL 0 *

INTEGER*4 804

INTEGER*4 844

612 \$PAGE

Line# 1 7

613 C

614 C

615 SUBROUTINE DRAW (V, N, CLR)

616 C

617 C

618 C-----

619 C

620 C

621 C This procedure actually draws the top view of TOM_B.

622 C

623 C This procedure must be modified if different hardware
624 C is used for the graphics display

625 C

626 C

627 C-----

628 C

629 C

630 REAL V(3, 10)

631 INTEGER X1, X2, Y1, Y2

632 INTEGER CLR

633 C

634 C *** draw mobile base

635 C

636 CALL RCT (V, 1, CLR)

637 C

638 C *** draw connecting line

639 C

640 X1 = V(1,1)

641 Y1 = V(2,1)

642 X2 = V(1,6)

643 Y2 = V(2,6)

644 CALL LINE (X1, Y1, X2, Y2, CLR)

645 C

646 C *** draw mocked-up

647 C

648 CALL RCT (V, 6, CLR)

649 C

650 CALL PURGE

651 CALL GRFRDY

652 CALL HOME

653 C

654 RETURN

655 END

Name Type Offset P Class

CLR INTEGER*4 8 *

I INTEGER*4 4 *

REAL 0 *

< O M V P L O T >>>

Page 22

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

Line#	1	7
INTEGER*4		892
INTEGER*4		900
INTEGER*4		896
INTEGER*4		904

656 \$PAGE

) Line# 1 7

657 C

658 C

659 SUBROUTINE CORDX (C, T, W)

660 C

661 C

662 C-----

663 C

664 C

665 C This procedure computes the necessary transformation

666 C matrices from floor to screen coordinates

667 C

668 C

669 C-----

670 C

671 C

672 INTEGER C(2,2)

673 REAL T(3,3), W(2)

674 C

675 C *** set up transformation matrix T

676 C

677 T(1,3) = C(1,1)

678 T(2,3) = C(2,2)

679 T(3,3) = 1.0

680 C

681 T(1,1) = (C(2,1) - T(1,3)) / W(1)

682 T(2,1) = (C(2,2) - T(2,3)) / W(1)

683 T(3,1) = 0.0

684 C

685 T(1,2) = (C(1,1) - T(1,3)) / W(2)

686 T(2,2) = (C(1,2) - T(2,3)) / W(2)

687 T(3,2) = 0.0

688 C

689 RETURN

690 END

ame Type Offset P Class

INTEGER*4 0 *

REAL 4 *

REAL 8 *

691 \$PAGE

<< O M V P L O T >>>

Page 24

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

692 C

693 C

694 C

695 C

696 C

697 C

698 C

699

SUBROUTINE PURGE

700 C

701 C

702 C

703 C

This procedure purges the graphics buffer and forces the board to complete the drawing by closing the graphics channel.

704

INTEGER GRF

705

CHARACTER ESC

706

COMMON /GMBD/ GRF, ESC

707

CLOSE (GRF)

708

RETURN

709

END

ne Type Offset P Class

PC CHAR*1 4 /GMBD /

F INTEGER*4 0 /GMBD /

710 \$PAGE

<<< O M V P L O T >>>

Page 25

07-14-84

14:03:20

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

711 C

712 C

713 SUBROUTINE GRFRDY

714 C

715 C This procedure opens the graphics channel and sets it ready for
716 C communication

717 C

718 C

719 INTEGER GRF

720 CHARACTER ESC

721 COMMON /GMBD/ GRF, ESC

722 OPEN (GRF, FILE = 'gm')

723 RETURN

724 END

Name	Type	Offset	P	Class
ESC	CHAR*1	4	/GMBD	/
GRF	INTEGER*4	0	/GMBD	/

725 \$PAGE

OMV PLOT >>>

Page 26

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

726 C

727 C

728

SUBROUTINE SETFB (FG, BG)

729 C

730 C

731 C

This procedure sets the foreground color to FG and the background color to BG. Both arguments must be of INTEGER type.

732 C

733 C

734 C

735

INTEGER GRF, FG, BG

736

CHARACTER ESC

737

COMMON /GMBD/ GRF, ESC

738

WRITE (GRF, 10) ESC, FG, BG

739

RETURN

740 10

FORMAT (' ', A1, '!', I2, ';', I2, 'c\\')

741

END

ame Type Offset P Class

INTEGER*4 4 *

SC CHAR*1 4 /GMBD /

G INTEGER*4 0 *

F INTEGER*4 0 /GMBD /

742 \$PAGE

<< O M V P L O T >>>

Page 27

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

D Line# 1 7

743 C

744 C

745 SUBROUTINE GRAFICS

746 C

747 C

748 C This procedure enters the GM graphics mode with a four-line text

749 C window at the bottom

750 C

751 C

752 INTEGER GRF

753 CHARACTER ESC

754 COMMON /GMBD/ GRF, ESC

755 C

756 GRF = 9

757 ESC = CHAR(27)

758 CALL GRFRDY

759 WRITE (GRF, 10) ESC

760 WRITE (GRF, 20) ESC

761 C WRITE (GRF, 30) ESC

762 CALL SETFB (1, 0)

763 CALL HOME

764 RETURN

765 C

766 10 FORMAT (' ', A1, '(!0m'\)

767 20 FORMAT (' ', A1, '(!640;352;2g'\)

768 30 FORMAT (' ', A1, '(!21;24r'\)

769 END

Name	Type	Offset	P	Class
------	------	--------	---	-------

-CHAR				INTRINSIC
ESC	CHAR*1	4	/GMBD	/
GRF	INTEGER*4	0	/GMBD	/

770 \$PAGE

```
Line# 1      7
771 C
772 C
773 SUBROUTINE QUITGM
774 C
775 C
776 C      This procedure gets one out of graphics mode and returns
777 C      to text mode
778 C
779 C
780 CHARACTER      CH, ESC
781 INTEGER         GRF
782 COMMON /GMBD/ GRF, ESC
783 C
784 CALL HOME
785 WRITE (GRF, 30)
786 CALL PURGE
787 READ (*, 10) CH
788 CALL GRFRDY
789 CALL TEXT
790 RETURN
791 10  FORMAT (A1)
792 30  FORMAT ('Press <CR> to continue ... '\)
793 END
```

me	Type	Offset	P	Class
	CHAR*1	1015		
C	CHAR*1	4	/GMBD	/
	INTEGER*4	0	/GMBD	/

- 794 \$PAGE

<<< O M V P L O T >>>

Page 29

07-14-84

14:03:20

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

795 C

796 C

797 SUBROUTINE TEXT

798 C

799 C

800 C This procedure returns the system to text mode

801 C

802 C

803 INTEGER GRF

804 CHARACTER ESC

805 COMMON /GMBD/ GRF, ESC

806 WRITE (GRF, 10) ESC

807 RETURN

808 C

809 10 FORMAT (' ', A1, '![80;25;1a'\)

810 END

Name	Type	Offset	P	Class
------	------	--------	---	-------

ESC	CHAR*1	4	/GMBD	/
-----	--------	---	-------	---

GRF	INTEGER*4	0	/GMBD	/
-----	-----------	---	-------	---

811 \$PAGE

<< O M V P L O T >>>

Page 30

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

812 C

813 C

814 SUBROUTINE LINE (X1, Y1, X2, Y2, COLOR)

815 C

816 C

817 C This procedure draws a line from (X1,Y1) to (X2,Y2) in COLOR

818 C

819 C

820 INTEGER GRF, X1, Y1, X2, Y2, COLOR

821 CHARACTER ESC

822 COMMON /GMBD/ GRF, ESC

823 WRITE (GRF, 10) ESC, X1, Y1, X2, Y2, COLOR

824 10 FORMAT (' ', A1, '!', 4(I3,';'), I3, '1\')

825 END

ie Type Offset P Class

DLOR	INTEGER*4	16 *	
	CHAR*1	4	/GMBD /
	INTEGER*4	0	/GMBD /
1	INTEGER*4	0 *	
	INTEGER*4	8 *	
	INTEGER*4	4 *	
2	INTEGER*4	12 *	

826 \$PAGE

```
D Line# 1      7
827 C
828 C
829      SUBROUTINE      HIDE LN (X1, Y1, X2, Y2, COLOR)
830 C
831 C
832 C      This procedure draws the line (X1,Y1) - (X2,Y2) but aborts drawi
833 C      before reaching target if a dot in a color other than BG is
834 C      encountered
835 C
836 C
837      INTEGER      GRF, X1, Y1, X2, Y2, COLOR
838      CHARACTER      ESC
839      COMMON      /GMBD/ GRF, ESC
840      WRITE (GRF, 10) ESC, X1, Y1, X2, Y2, COLOR
841      RETURN
842 10      FORMAT (' ', A1, ' [!', 4(I3, ';'), I3, 'S'\)
843      END
```

ame	Type	Offset	P	Class
COLOR	INTEGER*4	16	*	
SC	CHAR*1	4	/GMBD	/
GRF	INTEGER*4	0	/GMBD	/
X1	INTEGER*4	0	*	
Y2	INTEGER*4	8	*	
	INTEGER*4	4	*	
	INTEGER*4	12	*	

844 \$PAGE

line# 1 7

845 C

846 C

847

SUBROUTINE POLYGN (N, X, Y, COLOR)

848 C

849 C

850 C

This procedure draws a closed polygon whose N vertices are store
in the arrays X and Y. The color to be used is COLOR

851 C

852 C

853 C

854

INTEGER GRF, X(N), Y(N), COLOR

855

CHARACTER ESC

856

COMMON /GMBD/ GRF, ESC

857

WRITE (GRF, 10) ESC

858

DO 100 K = 1, N

859

WRITE (GRF, 20) X(K), Y(K)

860 100

CONTINUE

861

WRITE (GRF, 30) COLOR

862

RETURN

863 10

FORMAT (' ', A1, '![!'\)

864 20

FORMAT (' 2(I3, ';'')\)

865 30

FORMAT (' I3, 'p'\)

866

END

ame	Type	Offset	P	Class
LOR	INTEGER*4	12	*	
CC	CHAR*1	4	/GMBD	/
RF	INTEGER*4	0	/GMBD	/
	INTEGER*4	1160		
	INTEGER*4	0	*	
	INTEGER*4	4	*	
	INTEGER*4	8	*	

867 \$PAGE

<<< O M V P L O T >>>

Page 33

07-14-84

14:03:20

Microsoft FORTRAN77 V3.13 8/05/83

```
D Line# 1      7
      868 C
      869 C
      870      SUBROUTINE HOME
      871 C
      872 C
      873 C      THIS SUBROUTINE HOMES THE CURSOR
      874 C
      875 C
      876      INTEGER      GRF
      877      CHARACTER      ESC
      878 C
      879      COMMON /GMBD/ GRF, ESC
      880 C
      881      WRITE (GRF, 10) ESC
      882      RETURN
      883 10      FORMAT (' ', A1, '[ 1;1 f'\)
      884      END
```

Name	Type	Offset	P	Class
-ESC	CHAR*1	4	/GMBD	/
GRF	INTEGER*4	0	/GMBD	/

885 \$PAGE

< OMV PLOT >>>

Page 34

07-14-84

14:03:20

Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

Type	Size	Class
ORDX		SUBROUTINE
FLR		SUBROUTINE
IBD	5	SUBROUTINE
AFIC		COMMON
RDY		SUBROUTINE
ELN		SUBROUTINE
ME		SUBROUTINE
TPL		SUBROUTINE
E		SUBROUTINE
	12	COMMON
	312	COMMON
	28	COMMON
VE		SUBROUTINE
THING		SUBROUTINE
POS		SUBROUTINE
OT		SUBROUTINE
OLYGN		SUBROUTINE
GE		SUBROUTINE
TGM		SUBROUTINE
CT		SUBROUTINE
DRAW		SUBROUTINE
TFB		SUBROUTINE
DEVE		SUBROUTINE
INCOS		SUBROUTINE
XT		SUBROUTINE
PVEW		SUBROUTINE
MUL		SUBROUTINE

Pass One No Errors Detected
885 Source Lines

Appendix 4

OMV -- Data files

File : INITCON.DAT

This file contains all the needed initial conditions

0.0	POS(1) -- initial condition	
0.0	POS(2) -- initial condition	
0.0	POS(3) -- initial condition	
0.00	VEL(1) -- initial condition	
0.0	VEL(2) -- initial condition	
0.0	VEL(3) -- initial condition	
0.0	EUL(1) -- initial condition	.. ROLL
0.0	EUL(2) -- initial condition	.. PITCH
0.0	EUL(3) -- initial condition	.. YAW

File : MDLPRM.DAT

This file contains all the model parameters needed by OMV

00.075	ACC(1) : Acc along X-axis (body)
00.075	ACC(2) : Acc along Y-axis (body)
00.075	ACC(3) : Acc along Z-axis (body)
000.52359878	WWB(1) : body rate about X axis
000.52359878	WWB(2) : body rate about Y axis
000.52359878	WWB(3) : body rate about Z axis
7048.37	III(1) principal moment of inertia along 1 axis
3713.95	III(2) principal moment of inertia along 2 axis
3713.95	III(3) principal moment of inertia along 3 axis
3282.75	Mass in kilograms
0.1	major cycle period in seconds
1	MODE : 1 for position control
10	No. of steps per major cycle
200.0	altitude of orbit in kilo-meters

File : SVXINT.DAT

This file contains all the system initialization data needed by the SVX module

0.5588	CC	IN METERS
0.762	LL	IN METERS
11.668	AA	IN METERS
2.4384	HH	IN METERS
7048.37	IINV(1)	
3713.95	IINV(2)	
3713.95	IINV(3)	

File : HNDSGL.DAT

This file contains the simulated hand controller signals
(Partial list)

[illegible]

File : SIZE.DAT

This file contains all the plot parameters
for the
graphics package PLOT

0.5588	CC : 22 inches
2.1336	DD : 84 inches
0.762	LL : 30 inches
1.016	RR : 40 inches
0.6096	WW : 24 inches
0.3048	TT : 12 inches
409	CORNR(1,1)
001	CORNR(1,2)
630	CORNR(2,1)
350	CORNR(2,2)
100	CORNR(1,1) SIDE VIEW
152	CORNR(1,2) SIDE VIEW
500	CORNR(2,1) SIDE VIEW
300	CORNR(2,2) SIDE VIEW
000	PLOT MODE : <> 0 MEANS NO CLEAR
000	VIEW : 0 = TOP VIEW, <> 0 = SIDE VIEW
001	PRTFG: 1-PLOT 2-PRINT 3-PLOT & PRINT

Appendix 5

OMV -- Source Listing

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

1 \$LINESIZE:79
2 \$PAGESIZE: 56
3 \$TITLE: '<<< O M V >>>'

OMV SIMULATION MODEL

by

Dr. W. Teoh

U A H Huntsville
1984

This is a simplified version of a mathematical simulation model of the OMV. In this model, the following simplifications and assumptions are made :

1. The hand controllers provide signals that are interpreted as a force at the center of mass and/or a torque about the center of mass to provide a rotation of constant angular velocity.
2. The target vehicle is in a circular orbit; the altitude of this orbit is inputted from the MDLPRM.DAT file.
3. Orbital mechanics is implemented, but smaller perturbation effects are totally ignored.
4. Detailed placement of thrusters is not considered (Please see assumption 1. above)
5. Roll, pitch and yaw denote the instantaneous orientation of the OMV.

A 14 component state vector is generated by this model, and this state vector serves as input to the SVX module.

REAL * 8 X(3), V(3), E(3), A(3), W(3), Q(4)
REAL * 8 POS(3), VEL(3), EUL(3), OMEGA
REAL * 8 III(3), S(14), MASS, CYCLE
INTEGER CMD(7), IN, FLAG, MODE, STEP
INTEGER * 4 TIME

COMMON /MC/ III, MASS, CYCLE, MODE, STEP
COMMON /PC/ POS, VEL, EUL, OMEGA

```
D Line# 1      7
50 C      *** system initialization
51 C
52      IN = 2
53      TIME = -1
54      CALL OMVMDL (IN)
55      OPEN (IN, FILE = 'HNDSGL.DAT')
56 C
57 C      *** *** Note : this invokes graphics routines, and can be
58 C                  eliminated if no graphics output.
59 C
60      CALL INITPL
61 C
62 C      *** calculate the initial quaternions at the start of the
63 C      *** simulation and read hand controller
64 C
65      CALL DETQ (EUL, Q)
66      CALL HNDCTL (IN, FLAG, A, W)
67      CALL MATCH (EUL, POS, VEL, E, X, V, 3)
68      CALL STATE (Q, S, W)
69      CALL SVX (S, CMD, MODE)
70      CALL OUTPUT (A, W, X, V, E, Q, S, CMD, TIME)
71      TIME = 0
72 C
73 C      *** main processing loop
74 C
75 C      WHILE (FLAG = 0) DO
76 100          IF (FLAG .NE. 0) GOTO 900
77 C
78 C          *** copy initial state into work vectors and use these
79 C          *** work vectors for solving the equations of motion
80 C
81      CALL MOTION (X, V, E, A, W, Q)
82 C
83 C          *** update dynamic state
84 C
85      CALL MATCH (E, X, V, EUL, POS, VEL, 3)
86 C
87 C          *** calculate state vector and pass it on to the State
88 C          *** Vector Transformation module
89 C
90      CALL STATE (Q, S, W)
91      CALL SVX (S, CMD, MODE)
92      CALL OUTPUT (A, W, X, V, E, Q, S, CMD, TIME)
93 C
94 C          *** poll hand controller and get the next set of signals
95 C
96      CALL HNDCTL (IN, FLAG, A, W)
97      GOTO 100
98 C      END WHILE
```

< O M V >>>

Page 3
07-14-84
12:51:14

Line# 1 7 Microsoft FORTRAN77 V3.13 8/05/83
99 900 CONTINUE
100 CLOSE (IN)
101 C
102 C *** *** This is also a call to the graphics package
103 C
104 CALL QUITGM
105 C
106 C *** Grand exit, stage left
107 C
108 STOP
109 END

ame	Type	Offset	P	Class
	REAL*8	242		
	INTEGER*4	266		
YCLE	REAL*8	32	/MC	/
	REAL*8	74		
	REAL*8	48	/PC	/
LAG	INTEGER*4	302		
II	REAL*8	0	/MC	/
	INTEGER*4	294		
ASS	REAL*8	24	/MC	/
ODE	INTEGER*4	40	/MC	/
EGA	REAL*8	72	/PC	/
S	REAL*8	0	/PC	/
	REAL*8	98		
	REAL*8	130		
EP	INTEGER*4	44	/MC	/
IME	INTEGER*4	298		
	REAL*8	26		
L	REAL*8	24	/PC	/
	REAL*8	50		
	REAL*8	2		

110 \$PAGE

D Line# 1 7

111 C

112 C

113 SUBROUTINE OMVMDL (IN)

114 C

115 C-----

116 C

117 C

118 C

119 C

120 C

121 C

122 C-----

123 C

124

REAL * 8 POS(3), VEL(3), EUL(3), OMEGA

125

REAL * 8 ACC(3), III(3), WWB(3), INV(3)

126

REAL * 8 MASS, CYCLE, ORBIT

127

INTEGER IN, MODE, STEP

128 C

129

COMMON /DC/ ACC, WWB

130

COMMON /MC/ III, MASS, CYCLE, MODE, STEP

131

COMMON /PC/ POS, VEL, EUL, OMEGA

132 C

133 C

*** get initial conditions of the OMV

134 C

135

OPEN (IN, FILE = 'INITCON.DAT')

136

CALL VECTOR (IN, POS, 3)

137

CALL VECTOR (IN, VEL, 3)

138

CALL VECTOR (IN, EUL, 3)

139

CLOSE (IN)

140 C

141 C

*** read acceleration, angular rates and

142 C

*** principal moments of inertia in body frame

143 C

144

OPEN (IN, FILE = 'MDLPRM.DAT')

145

CALL VECTOR (IN, ACC, 3)

146

CALL VECTOR (IN, WWB, 3)

147

CALL VECTOR (IN, III, 3)

148 C

149 C

*** read mass characteristics & other parameters

150 C

151

READ (IN, 10) MASS

152

READ (IN, 10) CYCLE

153

READ (IN, 20) MODE

154

READ (IN, 30) STEP

155

READ (IN, 10) ORBIT

156

CLOSE (IN)

157 C

158 C

*** calculate orbital frequency

159 C

<< O M V >>>

Page 5
07-14-84
12:51:14

Line# 1 7
160 CALL ANGFRE (ORBIT, OMEGA)
161 C
162 C
163 RETURN
164 10 FORMAT (F15.8)
165 20 FORMAT (I1)
166 30 FORMAT (I2)
167 END

Microsoft FORTRAN77 V3.13 8/05/83

ame	Type	Offset	P	Class
	REAL*8	0	/DC	/
ANGLE	REAL*8	32	/MC	/
UL	REAL*8	48	/PC	/
	REAL*8	0	/MC	/
	INTEGER*4	0	*	
NV	REAL*8	306		
ASS	REAL*8	24	/MC	/
DE	INTEGER*4	40	/MC	/
MEGA	REAL*8	72	/PC	/
ORBIT	REAL*8	330		
S	REAL*8	0	/PC	/
LEP	INTEGER*4	44	/MC	/
EL	REAL*8	24	/PC	/
B	REAL*8	24	/DC	/

- 168 \$PAGE

<<< O M V >>>

Page 6

07-14-84

12:51:14

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

169 C

170 C

171 SUBROUTINE ANGFRE(ORB, W)

172 C

173 C

174 C-----

175 C

176 C This procedure calculates the orbital angular frequency

177 C at a given altitude. In this calculation, the altitude

178 C must be given in kilo-meters. This is necessary in order

179 C for the calculations to be carried out without lossing

180 C precision. The angular frequency W is in rad/second

181 C

182 C-----

183 C

184 REAL * 8 ORB

185 REAL * 8 ALT, R3, W

186 C

187 ALT = ORB * 0.001

188 R3 = (6.370 + ALT) ** 3

189 W = DSQRT (398.86 / R3) * 0.001

190 RETURN

191 END

Name Type Offset P Class

ALT REAL*8

358

-DSQRT INTRINSIC

ORB REAL*8

0 *

R3 REAL*8

366

W REAL*8

4 *

192 \$PAGE

O M V >>>

Page 7

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

193 C

194 C

195 SUBROUTINE VECTOR (M, A, N)

196 C

197 C

198 C-----

199 C

200 C

201 C This procedure reads a vector A of N elements from input

202 C unit M

203 C

204 C-----

205 C

206 INTEGER M, N

207 REAL * 8 A(N)

208 C

209 DO 100 K = 1, N

210 READ (M, 10) A(K)

211 100 CONTINUE

212 RETURN

213 10 FORMAT (F15.8)

214 END

me Type Offset P Class

REAL*8 4 *

INTEGER*4 374

INTEGER*4 0 *

INTEGER*4 8 *

215 \$PAGE

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

216 C

217 C

218 SUBROUTINE HNDCTL (IN, FLAG, A, W)

219 C

220 C-----

221 C

222 C Simulates hand controllers input by reading from a file
223 C (called HNDSGL.DAT 12) integers to simulate a 12 bit output
224 C of the hand controllers. Bit assignments are as follows :

225 C

226 C bit meaning (direction in body frame)

227 C ===

228 C 1 Accelerate along +1 axis

229 C 2 Accelerate along -1 axis

230 C 3 Accelerate along +2 axis

231 C 4 Accelerate along -2 axis

232 C 5 Accelerate along +3 axis

233 C 6 Accelerate along -3 axis

234 C 7 Rotate about +1 axis

235 C 8 Rotate about -1 axis

236 C 9 Rotate about +2 axis

237 C 10 Rotate about -2 axis

238 C 11 Rotate about +3 axis

239 C 12 Rotate about -3 axis

240 C

241 C-----

242 C

243 REAL * 8 ACC(3), WWB(3)

244 REAL * 8 A(3), W(3)

245 INTEGER SL(6), SA(6), FLAG

246 COMMON /DC/ ACC, WWB

247 C

248 FLAG = 0

249 READ (IN, 10, END = 90, ERR = 90) SL, SA

250 C

251 C *** no error, generate matrices A and W

252 C

253 CALL FUDGE (A, ACC, SL)

254 CALL FUDGE (W, WWB, SA)

255 RETURN

256 90 CONTINUE

257 C

258 C *** error condition

259 C

260 FLAG = 1

261 RETURN

262 10 FORMAT (20I1)

263 END

< O M V >>>

Page 9

07-14-84

12:51:14

Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

e	Type	Offset	P	Class
-	REAL*8	8	*	
	REAL*8	0	/DC	/
AG	INTEGER*4	4	*	
-	INTEGER*4	0	*	
	INTEGER*4	414		
	INTEGER*4	390		
	REAL*8	12	*	
7	REAL*8	24	/DC	/

- 264 \$PAGE

<<< O M V >>>

Page 10

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

D Line# 1 7

265 C

266 C

267 SUBROUTINE FUDGE (A, ACC, SL)

268 C

269 C-----

270 C

271 C *** Sets appropriate components based on SL

272 C

273 C-----

274 C

275 INTEGER SL(6), T, K, J

276 REAL * 8 ACC(3), A(3), X

277 DO 100 K = 1, 6, 2

278 J = (K + 1) / 2

279 X = 0.0

280 T = SL(K) + SL(K+1)

281 IF (T .EQ. 1) THEN

282 X = ACC(J)

283 IF (SL(K) .EQ. 0) X = -X

284 END IF

285 A(J) = X

286 100 CONTINUE

287 RETURN

288 END

Name	Type	Offset	P	Class
------	------	--------	---	-------

-A	REAL*8	0	*	
ACC	REAL*8	4	*	
J	INTEGER*4	450		
-K	INTEGER*4	446		
SL	INTEGER*4	8	*	
T	INTEGER*4	462		
X	REAL*8	454		

289 \$PAGE

Line# 1 7

SUBROUTINE STATE (Q, S, W)

This procedure uses the dynamic quantities of the OMV and constructs a State Vector of the OMV. The 14 components of this State Vector S are defined as follows

Components	Meaning
S(1) .. S(3)	Relative displacement between OMV and target
S(4) .. S(6)	Relative velocity between OMV & target
S(7) .. S(9)	Angular momentum vector of OMV in LVF
S(10) .. S(13)	Attitude quaternions expressed in body frame, and
S(14)	Instantaneous mass, assumed constant throughout the simulation.

```

REAL * 8      POS(3), VEL(3), EUL(3), OMEGA
REAL * 8      III(3), QQ(4), MASS,  CYCLE
REAL * 8      LB(3), LL(3), B(3,3), A(3,3)
REAL * 8      Q(4),  W(3),  L(3),  S(14)
INTEGER      MODE, STEP

```

```

COMMON /MC/   III, MASS, CYCLE, MODE, STEP
COMMON /PC/   POS, VEL, EUL, OMEGA

```

```

*** calculate angular momentum in body frame

```

```

CALL DOTPRD (III, W, LB, 3)

```

```

*** transforms it to local vertical frame

```

```

CALL DCSINV (Q, B)
CALL DMUL  (B, LB, LL, 3)

```

```

*** Build state vector

```

```

N = 0
CALL PUT (N, S, POS, 3)
CALL PUT (N, S, VEL, 3)
CALL PUT (N, S, LL , 3)
CALL PUT (N, S, Q  , 4)

```

<<< O M V >>>

Page 12

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

D Line# 1 7
339 S(14) = MASS
340 C
341 RETURN
342 END

Name	Type	Offset	P	Class
I	REAL*8	642		
J	REAL*8	570		
CYCLE	REAL*8	32	/MC	/
TUL	REAL*8	48	/PC	/
II	REAL*8	0	/MC	/
L	REAL*8	546		
B	REAL*8	498		
L	REAL*8	522		
MASS	REAL*8	24	/MC	/
MODE	INTEGER*4	40	/MC	/
	INTEGER*4	714		
MEGA	REAL*8	72	/PC	/
POS	REAL*8	0	/PC	/
	REAL*8	0	*	
Q	REAL*8	466		
S	REAL*8	4	*	
STEP	INTEGER*4	44	/MC	/
EL	REAL*8	24	/PC	/
w	REAL*8	8	*	

343 \$PAGE

O M V >>>

Page 13

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

344 C

345 C

346 SUBROUTINE PUT (N, S, A, M)

347 C

348 C

349 C

350 C *** The procedure copies a vector A into a larger one S
351 C starting at the N-th element of S

352 C

353 C

354 C

355 REAL * 8 S(14)

356 REAL * 8 A(M)

357 C

358 DO 100 K = 1, M

359 N = N + 1

360 S(N) = A(K)

361 100 CONTINUE

362 RETURN

363 END

me Type Offset P Class

REAL*8 8 *

INTEGER*4 718

INTEGER*4 12 *

INTEGER*4 0 *

REAL*8 4 *

364 \$PAGE

<<< O M V >>>

Page 14

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

D Line# 1 7

365 C

366 C

367 SUBROUTINE DOTPRD (A, B, C, N)

368 C

369 C-----

370 C

371 C *** This procedure calculates a vector C from two other

372 C vectors A and B such that

373 C C(I) = A(I) * B(I)

374 C for all i = 1 to N

375 C

376 C-----

377 C

378 REAL * 8 A(N), B(N), C(N)

379 DO 100 K = 1, N

1 380 C(K) = A(K) * B(K)

1 381 100 CONTINUE

382 RETURN

383 END

Name Type Offset P Class

A REAL*8 0 *

-B REAL*8 4 *

C REAL*8 8 *

K INTEGER*4 726

N INTEGER*4 12 *

384 \$PAGE

<<< O M V >>>

Page 15

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

385 C

386 C

387 SUBROUTINE DETQ (E, Q)

388 C

389 C-----

390 C

391 C *** calculates quaternions from the Euler angles

392 C using expression given by Zack.

393 C

394 C-----

395 C

396 REAL * 8 E(3), Q(4)

397 REAL * 8 C1, S1, C2, S2, C3, S3, THETA

398 C

399 THETA = E(1) / 2.0

400 CALL SINCOS (THETA, S1, C1)

401 THETA = E(2) / 2.0

402 CALL SINCOS (THETA, S2, C2)

403 THETA = E(3) / 2.0

404 CALL SINCOS (THETA, S3, C3)

405 C

406 Q(1) = S1 * C3 * C2 + C1 * S3 * S2

407 Q(2) = S1 * S3 * C2 + C1 * C3 * S2

408 Q(3) = C1 * S3 * C2 - S1 * C3 * S2

409 Q(4) = C1 * C3 * C2 - S1 * S3 * S2

410 C

411 RETURN

412 END

Name Type Offset P Class

1 REAL*8 750

2 REAL*8 766

3 REAL*8 782

4 REAL*8 0 *

5 REAL*8 4 *

61 REAL*8 742

62 REAL*8 758

63 REAL*8 774

THETA REAL*8 734

413 \$PAGE

<<< O M V >>>

Page 16

07-14-84

12:51:14

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

414 C

415 C

416 SUBROUTINE SINCOS (THETA, S, C)

417 C

418 C

419 C-----

420 C

421 C *** this procedure returns the sine and cosine of an
422 C angle THETA.

423 C

424 C-----

425 C

426 REAL * 8 THETA, S, C, A

427 C

428 C = DCOS(THETA)

429 S = DSIN(THETA)

430 RETURN

431 END

Name	Type	Offset	P	Class
A	REAL*8	*****		
C	REAL*8	8 *		
DCOS				INTRINSIC
DSIN				INTRINSIC
S	REAL*8	4 *		
THETA	REAL*8	0 *		

432 \$PAGE


```
Line# 1      7
433 C
434 C
435      SUBROUTINE MOTION (X, V, E, A, W, Q)
436 C
437 C-----
438 C
439 C      *** This procedure solves the equation of motion
440 C
441 C-----
442 C
443      REAL * 8      POS(3), VEL(3), EUL(3), OMEGA
444      REAL * 8      X(3), V(3), E(3), A(3), W(3), Q(4)
445      REAL * 8      CIN(3,3), C(3,3), AA(3,10), B(3), QQ(4)
446      REAL * 8      WW(3), PI, TWO
447      REAL * 8      III(3), MASS, CYCLE
448      INTEGER      MODE, STEP
449 C
450      COMMON /MC/    III, MASS, CYCLE, MODE, STEP
451      COMMON /PC/    POS, VEL, EUL, OMEGA
452 C
453      H = CYCLE / FLOAT(STEP)
454      N = STEP
455      PI = 355.0 / 113.0
456      TWO= PI * 2.0
457 C
458 C      *** Divide 1 major cycle into N equal subintervals and
459 C      *** determine the OMV state for each interval
460 C
461      DO 100 KK = 1, N
462 C
463 C      *** Update orientation
464 C
465      DO 200 J = 1, 3
466          WW(J) = W(J) * H
467          E(J) = E(J) + WW(J)
468          IF (E(J) .GT. TWO) E(J) = E(J) - TWO
469 200 CONTINUE
470 C
471 C      *** Calculate quaternion for this rotation, and transform
472 C      *** it to local vertical frame with respect to initial frame
473 C
474      CALL DETQ( WW, QQ)
475      CALL UPDQ (Q, QQ)
476 C
477 C      *** from the direction cosine matrix, calculate the
478 C      *** acceleration vector in LVF and store it in the
479 C      *** acceleration matrix AA
480 C
481      CALL DCSINV (Q, CIN)
```

<<< O M V >>>

Page 18

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

```
D Line# 1      7
1 482          CALL DMUL (CIN, A, B, 3)
1 483          CALL STORE (B, AA, KK)
1 484 100      CONTINUE
485 C
486 C          *** Solve the equation of motion using the Adam-Brashford
487 C          *** method
488 C
489          CALL SOLVE (X, V, AA, N, H, OMEGA)
490 C
491          RETURN
492          END
```

Name	Type	Offset	P	Class
A	REAL*8	12	*	
AA	REAL*8	990		
B	REAL*8	1230		
C	REAL*8	918		
CIN	REAL*8	846		
CYCLE	REAL*8	32	/MC	/
E	REAL*8	8	*	
EUL	REAL*8	48	/PC	/
FLOAT				INTRINSIC
H	REAL	1254		
III	REAL*8	0	/MC	/
J	INTEGER*4	1286		
KK	INTEGER*4	1278		
MASS	REAL*8	24	/MC	/
MODE	INTEGER*4	40	/MC	/
N	INTEGER*4	1258		
OMEGA	REAL*8	72	/PC	/
PI	REAL*8	1262		
POS	REAL*8	0	/PC	/
Q	REAL*8	20	*	
Q	REAL*8	814		
STEP	INTEGER*4	44	/MC	/
TWO	REAL*8	1270		
V	REAL*8	4	*	
VEL	REAL*8	24	/PC	/
W	REAL*8	16	*	
WW	REAL*8	790		
X	REAL*8	0	*	

493 \$PAGE

<< O M V >>>

Page 19

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

494 C

495 C

496 SUBROUTINE MATCH (A, B, C, P, Q, R, N)

497 C

498 C

499 C-----

500 C

501 C *** This procedure makes an exact duplicate B of a
502 C vector A of N elements

503 C

504 C-----

505 C

506 REAL * 8 A(N), B(N), C(N), P(N), Q(N), R(N)

507 DO 100 K = 1, 3

508 P(K) = A(K)

509 Q(K) = B(K)

510 R(K) = C(K)

511 100 CONTINUE

512 RETURN

513 END

me Type Offset P Class

REAL*8 0 *

REAL*8 4 *

REAL*8 8 *

INTEGER*4 1290

INTEGER*4 24 *

REAL*8 12 *

REAL*8 16 *

REAL*8 20 *

514 \$PAGE

<<< O M V >>>

Page 20

07-14-84

12:51:14

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

515 C

516 C

517 SUBROUTINE STORE (AAA, AA, K)

518 C

519 C

520 C

521 C

522 C This procedure takes an instantaneous acceleration vector

523 C AAA and stores it in the acceleration matrix AA which is needed

524 C by the numerical integration process

525 C

526 C

527 C

528 REAL * 8 AA(3, 10)

529 REAL * 8 AAA(3)

530 DO 100 J = 1, 3

531 AA(J,K) = AAA(J)

532 100 CONTINUE

533 RETURN

534 END

Name	Type	Offset	P	Class
------	------	--------	---	-------

AA	REAL*8	4	*	
----	--------	---	---	--

AAA	REAL*8	0	*	
-----	--------	---	---	--

J	INTEGER*4	1294		
---	-----------	------	--	--

K	INTEGER*4	8	*	
---	-----------	---	---	--

535 \$PAGE

Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

```
536 C
537 C
538 SUBROUTINE SOLVE(X,V,A,N,H,W)
539 C
540 C-----
541 C
542 C      This subroutine produces the numerical solution to the
543 C      system of equations of motion using a 3 step Adam-Brashford
544 C      method.
545 C
546 C-----
547 C
548 LOGICAL FLAG
549 REAL*8 X(3), V(3), A(3,10), AA(3,13), U(6,13)
550 REAL*8 WX2, WXW, WXWX3, HD12, F, W
551 COMMON /BLOCK/ AA, U, WX2, WXW, WXWX3, HD12
552 DATA FLAG /.TRUE./
553 C
554 C      *** pack user supplied nonhomomgenous part of DE
555 C      *** into the higher part of AA
556 C
557 DO 10 I = 1,10
558 DO 10 K = 1,3
559 AA(K,I+3) = A(K,I)
560 10 CONTINUE
561 C
562 C      *** if this is the first call to solve (FLAG = T), then
563 C      *** it is necessary to initialize some parameters
564 C
565 IF (FLAG) THEN
566 CALL INNIT(X,V,W,H)
567 FLAG = .FALSE.
568 END IF
569 C
570 C      *** use the Adams-Brashford 3-step method to advance the
571 C      *** solution H time units. Place the solution back into
572 C      *** X and V.
573 C
574 DO 100 I = 4,N+3
575 DO 100 J = 1,6
576 U(J,I) = U(J,I-1) +
577 + HD12*(23*F(J,I-1)-16*F(J,I-2)+5*F(J,I-3))
578 100 CONTINUE
579 X(1) = U(1,N+3)
580 V(1) = U(2,N+3)
581 X(2) = U(3,N+3)
582 V(2) = U(4,N+3)
583 X(3) = U(5,N+3)
584 V(3) = U(6,N+3)
```

<<< O M V >>>

Page 22

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

D Line# 1 7

585 C

586 C

587 C

588

DO 200 J = 1,6

1 589

DO 200 I = 1,3

2 590

U(J,I) = U(J,N+I)

2 591

IF (J .LE. 3) AA(I,J) = AA(I,N+J)

2 592 200

CONTINUE

593

RETURN

594

END

Name	Type	Offset	P	Class
A	REAL*8	8	*	
AA	REAL*8	0		/BLOCK /
F	REAL*8			FUNCTION
FLAG	LOGICAL*4	1298		
H	REAL	16	*	
HD12	REAL*8	960		/BLOCK /
I	INTEGER*4	1302		
J	INTEGER*4	1314		
K	INTEGER*4	1306		
N	INTEGER*4	12	*	
U	REAL*8	312		/BLOCK /
V	REAL*8	4	*	
W	REAL*8	20	*	
WX2	REAL*8	936		/BLOCK /
WXW	REAL*8	944		/BLOCK /
WXWX3	REAL*8	952		/BLOCK /
X	REAL*8	0	*	

595 \$PAGE

Line# 1 7

596 C

597 C

598

SUBROUTINE INNIT(X,V,W,H)

599 C

600 C

601 C

602 C

603 C

This procedure initializes all the necessary parameters
before solving the system of ordinary differential equations.
This procedure is invoked only once.

604 C

605 C

606 C

607 C

608 C

609

REAL * 8 X(3), V(3), AA(3,13), U(6,13), WX2, WXW, WXWX3

610

REAL * 8 CWT, SWT, T, W, HD12

611

COMMON /BLOCK/ AA, U, WX2, WXW, WXWX3, HD12

612

WXW = W*W

613

WXWX3 = 3*WXW

614

WX2 = 2*W

615

HD12 = DBLE(H)/12.0

616 C

617

DO 100 K = 1,3

618

U(2*K-1,3) = X(K)

619

U(2*K,3) = V(K)

620

DO 100 J = 1,6

621

AA(J,K) = 0.0

622 C

CONTINUE

623 100

CONTINUE

624 C

625

DO 300 I = 1,2

626

T = H*(I-3)

627

CWT = DCOS(W*T)

628

SWT = DSIN(W*T)

629

U(1,I) = X(1) + V(1)*(4*SWT-3*W*T)/W +

630

+ 6*X(3)*(SWT-W*T) + 2*V(3)*(CWT-1.0)/W

631

U(2,I) = V(1)*(4*CWT-3.0) + 6*W*X(3)*(CWT-1.0) -

632

+ 2*V(3)*SWT

633

U(3,I) = X(2)*CWT + V(2)*SWT/W

634

U(4,I) = -X(2)*W*SWT + V(2)*CWT

635

U(5,I) = 2*V(1)*(1.0-CWT)/W + X(3)*(4.0-3*CWT) +

636

+ V(3)*SWT/W

637

U(6,I) = 2*V(1)*SWT + 3*X(3)*W*SWT + V(3)*CWT

638 300

CONTINUE

639

RETURN

640

END

Name Type Offset P Class

A REAL*8 0 /BLOCK /

<<< O M V >>>

Page 24

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

D Line# 1 7

CWT	REAL*8	1362	
DBLE			INTRINSIC
DCOS			INTRINSIC
DSIN			INTRINSIC
I	REAL	12 *	
ID12	REAL*8	960	/BLOCK /
I	INTEGER*4	1350	
-J	INTEGER*4	1346	
(INTEGER*4	1342	
SWT	REAL*8	1370	
T	REAL*8	1354	
J	REAL*8	312	/BLOCK /
/	REAL*8	4 *	
W	REAL*8	8 *	
VX2	REAL*8	936	/BLOCK /
VXW	REAL*8	944	/BLOCK /
WXWX3	REAL*8	952	/BLOCK /
-X	REAL*8	0 *	

641 \$PAGE

<<< O M V >>>

Page 25

07-14-84

12:51:14

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

```

642 C
643 C
644      FUNCTION F(J,I)
645 C
646 C
647 C-----
648 C
649 C
650      REAL*8 AA(3,13),U(6,13),WX2,WXW,WXWX3,HD12,F
651      COMMON /BLOCK/ AA,U,WX2,WXW,WXWX3,HD12
652 C
653      GO TO (10,20,30,40,50,60), J
654 10      CONTINUE
655          F = U(2,I)
656          RETURN
657 20      CONTINUE
658          F = -WX2*U(6,I) + AA(1,I)
659          RETURN
660 30      CONTINUE
661          F = U(4,I)
662          RETURN
663 40      CONTINUE
664          F = -WXW*U(3,I) + AA(2,I)
665          RETURN
666 50      CONTINUE
667          F = U(6,I)
668          RETURN
669 60      CONTINUE
670          F = WX2*U(2,I) + WXWX3*U(5,I) + AA(3,I)
671          RETURN
672      END

```

Name	Type	Offset	P	Class
AA	REAL*8	0		/BLOCK /
HD12	REAL*8	960		/BLOCK /
I	INTEGER*4	4	*	
J	INTEGER*4	0	*	
U	REAL*8	312		/BLOCK /
WX2	REAL*8	936		/BLOCK /
WXW	REAL*8	944		/BLOCK /
WXWX3	REAL*8	952		/BLOCK /

673 \$PAGE

line# 1 7
 674 C
 675 C
 676 SUBROUTINE OUTPUT (A, W, X, V, E, Q, S, CMD, TIME)
 677 C
 678 C
 679 C-----
 680 C
 681 C This is the output section of the system. Any further
 682 C modification of the output requirements of this model must
 683 C be done in this procedure. In particular, if no output to
 684 C the CRT or printer is needed, it is recommended that C's
 685 C be inserted into column 1 of all the WRITE statements. The
 686 C simulation clock is updated in this procedure.
 687 C
 688 C-----
 689 C
 690 REAL * 8 A(3), W(3), X(3), V(3), E(3), Q(4), S(14)
 691 INTEGER CMD(7), EF, EEF, PRTFG
 692 INTEGER * 4 TIME, T
 693 C
 694 COMMON /ME/ EF, EEF, PRTFG
 695 C
 696 TIME = TIME + 1
 697 T = (TIME / 10) * 10 - TIME
 698 IF ((T .NE. 0) .OR. (PRTFG .EQ. 0)) RETURN
 699 IF (PRTFG .EQ. 1) GO TO 100
 700 OPEN (4, FILE = 'LPT1:')
 701 WRITE (4, 15) TIME / 10
 702 C WRITE (4, 10) A, W
 703 WRITE (4, 20) X, V
 704 WRITE (4, 30) E, W
 705 WRITE (4, 40) S
 706 WRITE (4, 50) CMD
 707 WRITE (4, 90)
 708 CLOSE (4)
 709 100 IF (PRTFG .NE. 2) CALL PLOT (CMD)
 710 C
 711 RETURN
 712 10 FORMAT (' A, W =', 3F10.6, 3X, 3F10.6)
 713 12 FORMAT (' ', 7I10)
 714 15 FORMAT (' TIME =', I6, ' Seconds')
 715 20 FORMAT (' X, V =', 3F10.6, 3X, 3F10.6)
 716 30 FORMAT (' E, W =', 3F10.6, 3X, 3F10.6/)
 717 40 FORMAT (' S =', 3F10.6, 3X, 3F10.6/
 718 1 ' ', 3F10.3/
 719 2 ' ', 4F10.6, 3X, F10.3/)
 720 50 FORMAT (' CMD =', 7I10)
 721 90 FORMAT (1H0)
 722 END

<<< O M V >>>

Page 27

07-14-84

12:51:14

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

Name	Type	Offset	P	Class
A	REAL*8	0	*	
CMD	INTEGER*4	28	*	
E	REAL*8	16	*	
EEF	INTEGER*4	4	/ME	/
EF	INTEGER*4	0	/ME	/
PRTFG	INTEGER*4	8	/ME	/
Q	REAL*8	20	*	
S	REAL*8	24	*	
T	INTEGER*4	1378		
TIME	INTEGER*4	32	*	
V	REAL*8	12	*	
W	REAL*8	4	*	
X	REAL*8	8	*	

723 \$PAGE

O M V >>>

Page 28

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

724 C

725 C

726

SUBROUTINE DMUL (A, B, C, N)

727 C

728 C

729 C

730 C

This procedure performs a matrix multiplication of an NxN
matrix A to an N-element column matrix B to yield an N-element
column matrix C

731 C

732 C

733 C

734 C

735 C

736

REAL * 8 A(N,N), B(N), C(N), S

737 C

738

DO 100 I = 1, N

739

S = 0.0

740

DO 200 J = 1, N

741

S = S + A(I,J) * B(J)

742 200

CONTINUE

743

C(I) = S

744 100

CONTINUE

745

RETURN

746

END

ie Type Offset P Class

REAL*8 0 *

REAL*8 4 *

REAL*8 8 *

INTEGER*4 1714

INTEGER*4 1730

INTEGER*4 12 *

REAL*8 1722

747 \$PAGE

<<< O M V >>>

Page 29

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

D Line# 1 7

748 C

749 C

750

SUBROUTINE UPDQ (Q, QQ)

751 C

752 C

753 C

754 C

755 C

This subroutine uses the previous quaternion and generates the present quaternions with respect to the local vertical frame LVF. Quaternion algebra is used to deduce the needed computation before hand to simplify the algorithm

756 C

757 C

758 C

759 C

760 C

761 C

762 C

763 C

764

REAL * 8 Q(4), QQ(4), Q1, Q2, Q3, Q4

765 C

766

Q1 = Q(1)*QQ(4) + Q(4)*QQ(1) - Q(3)*QQ(2) + Q(2)*QQ(3)

767

Q2 = Q(2)*QQ(4) + Q(3)*QQ(1) + Q(4)*QQ(2) - Q(1)*QQ(3)

768

Q3 = Q(3)*QQ(4) - Q(2)*QQ(1) + Q(1)*QQ(2) + Q(4)*QQ(3)

769

Q4 = Q(4)*QQ(4) - Q(1)*QQ(1) - Q(2)*QQ(2) - Q(3)*QQ(3)

770 C

771

Q(1) = Q1

772

Q(2) = Q2

773

Q(3) = Q3

774

Q(4) = Q4

775

RETURN

776

END

-Name Type Offset P Class

Q REAL*8 0 *

Q1 REAL*8 1738

Q2 REAL*8 1746

Q3 REAL*8 1754

Q4 REAL*8 1762

QQ REAL*8 4 *

777 \$PAGE

< O M V >>>

Page 30

07-14-84

12:51:14

Microsoft FORTRAN77 V3.13 8/05/83

Line# 1 7

778 C
779 C
780 SUBROUTINE DCSINV (Q, C)

781 C
782 C
783 C-----
784 C
785 C This subroutine takes the attitude quaternion Q and returns
786 C the transpose of the direction cosine matrix
787 C

788 C-----
789 C
790 C

791 REAL * 8 Q(4), C(3,3)
792 REAL * 8 Q1, Q2, Q3, Q4
793 REAL * 8 Q11, Q22, Q33, Q44
794 REAL * 8 Q12, Q13, Q23
795 REAL * 8 Q14, Q24, Q34

796 C
797 Q1 = Q(1)
798 Q2 = Q(2)
799 Q3 = Q(3)
800 Q4 = Q(4)

801 C
802 Q11 = Q1 * Q1
803 Q22 = Q2 * Q2
804 Q33 = Q3 * Q3
805 Q44 = Q4 * Q4

806 C
807 Q12 = 2.0 * Q1 * Q2
808 Q13 = 2.0 * Q1 * Q3
809 Q23 = 2.0 * Q2 * Q3
810 Q14 = 2.0 * Q1 * Q4
811 Q24 = 2.0 * Q2 * Q4
812 Q34 = 2.0 * Q3 * Q4

813 C
814 C(1,1) = Q11 - Q22 - Q33 + Q44
815 C(2,2) = -Q11 + Q22 - Q33 + Q44
816 C(3,3) = -Q11 - Q22 + Q33 + Q44

817 C
818 C(1,2) = Q12 - Q34
819 C(2,1) = Q12 + Q34
820 C(1,3) = Q13 + Q24
821 C(3,1) = Q13 - Q24
822 C(2,3) = Q23 - Q14
823 C(3,2) = Q23 + Q14

824 RETURN
825 END

<<< O M V >>>

Page 31

07-14-84

12:51:14

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

Name	Type	Offset	P	Class
------	------	--------	---	-------

C	REAL*8	4	*	
---	--------	---	---	--

Q	REAL*8	0	*	
---	--------	---	---	--

Q1	REAL*8	1770		
----	--------	------	--	--

Q11	REAL*8	1802		
-----	--------	------	--	--

Q12	REAL*8	1834		
-----	--------	------	--	--

Q13	REAL*8	1842		
-----	--------	------	--	--

Q14	REAL*8	1858		
-----	--------	------	--	--

Q2	REAL*8	1778		
----	--------	------	--	--

Q22	REAL*8	1810		
-----	--------	------	--	--

Q23	REAL*8	1850		
-----	--------	------	--	--

Q24	REAL*8	1866		
-----	--------	------	--	--

Q3	REAL*8	1786		
----	--------	------	--	--

Q33	REAL*8	1818		
-----	--------	------	--	--

Q34	REAL*8	1874		
-----	--------	------	--	--

Q4	REAL*8	1794		
----	--------	------	--	--

Q44	REAL*8	1826		
-----	--------	------	--	--

826 \$PAGE

< O M V >>>

Page 32

07-14-84

12:51:14

Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

ie	Type	Size	Class
3FRE			SUBROUTINE
CK		968	COMMON
		48	COMMON
2SINV			SUBROUTINE
TQ			SUBROUTINE
JL			SUBROUTINE
OTPRD			SUBROUTINE
	REAL*8		FUNCTION
GE			SUBROUTINE
NDCTL			SUBROUTINE
ITPL			SUBROUTINE
VIT			SUBROUTINE
AIN			PROGRAM
ATCH			SUBROUTINE
		48	COMMON
		12	COMMON
OTION			SUBROUTINE
VMDL			SUBROUTINE
TPUT			SUBROUTINE
C		80	COMMON
OT			SUBROUTINE
T			SUBROUTINE
UITGM			SUBROUTINE
INCOS			SUBROUTINE
LVE			SUBROUTINE
ATE			SUBROUTINE
TORE			SUBROUTINE
X			SUBROUTINE
DQ			SUBROUTINE
ECTOR			SUBROUTINE

Pass One No Errors Detected
826 Source Lines

Appendix 6

ADAM -- Source Listing

Line# 1 7

1 \$PAGESIZE : 56

2 \$TITLE: ' << A D A M >>'

3 C

4 C

5 C

6 C

7 C

8 C

9 C

10 C

11 C

12 C

13 C

14 C

15 C

16 C

17 C

18 C

19 C

20 C

21 C

22 C

23 C

24 C

25

26

27

28

29 C

30 C

31 C

32

33

34 C

35 C

36 C

37

38 C

39

40

41

42 100

43 C

44

45

46 C

47 C

48 C

49

Program : A D A M

by

Dr. W. Teoh

This program uses the Adam Brashforth method to solve the equation of motion (homogeneous case) numerically and compares the solution with the analytical results such that both outputs are printed.

REAL*8 XE(3),VE(3),X(3),V(3),A(3,10),W

REAL *8 XO(3), VO(3)

DATA A/30*0.0/

DATA N,H /10, 0.01/

WRITE (*, 30)

READ (*,32) W

get initial conditions

CALL GETINT (XO, VO, 3)

DO 100 K = 1, 3

X(K) = XO(K)

V(K) = VO(K)

CONTINUE

DO 10 I = 1,36000

T = 0.1*I

*** calculate the analytical solution

CALL EXACT(T,XE,VE,W,XO,VO)

D Line# 1 7

```

1 50 C
1 51 C      *** now get the numerical solution
1 52 C
1 53      CALL SOLVE(X,V,A,N,H,W)
1 54 C
1 55 C      *** output every 60 seconds
1 56 C
1 57      JJ = (I / 600) * 600
1 58      IF (JJ .EQ. I) THEN
1 59          WRITE(*,20) T,XE,VE
1 60          WRITE(*,20) T,X,V
1 61          WRITE (*, 22)
1 62      END IF
1 63 10      CONTINUE
1 64 C
1 65 20      FORMAT (F7.1, 6F12.6)
1 66 30      FORMAT (' ORBITAL RATE '\)
1 67 22      FORMAT (1H )
1 68 32      FORMAT (F15.8)
1 69      STOP
1 70      END

```

Name	Type	Offset P Class
-A	REAL*8	146
H	REAL	390
I	INTEGER*4	406
-JJ	INTEGER*4	414
K	INTEGER*4	402
N	INTEGER*4	386
T	REAL	410
-V	REAL*8	98
VO	REAL*8	122
VE	REAL*8	26
-W	REAL*8	394
X	REAL*8	50
XO	REAL*8	74
-XE	REAL*8	2

71 \$PAGE

Line# 1 7

SUBROUTINE EXACT(T,XE,VE,W,X,V)

** This subroutine calculates the exact solution
of the homogeneous ODEs

REAL*8 XE(3),VE(3),CWT,SWT,W, WT, X(3), V(3)

WT = W * T

SWT = DSIN(WT)

CWT = DCOS(WT)

XE(1) = X(1) + (4 * SWT - 3*WT)*V(1)/W + 6*(SWT - WT)*X(3)

+ 2 * (CWT - 1) * V(3) / W

XE(2) = CWT* X(2) + SWT * V(2) / W

XE(3) = 2 * (1 - CWT) * V(1) / W + (4 - 3 * CWT) * X(3)

- SWT * V(3) / W

VE(1) = (4 * CWT - 3) * V(1) + 6 * W * (CWT - 1) * X(3)

- 2 * SWT * V(3)

VE(2) = CWT * V(2) - W * SWT * X(2)

VE(3) = 2*SWT*V(1) + 3*W*SWT*X(3) + CWT*V(3)

RETURN

END

ime	Type	Offset	P	Class
WT	REAL*8	488		
DCOS				INTRINSIC
DSIN				INTRINSIC
SWT	REAL*8	480		
	REAL	0 *		
	REAL*8	20 *		
VE	REAL*8	8 *		
	REAL*8	12 *		
T	REAL*8	472		
X	REAL*8	16 *		
XE	REAL*8	4 *		

D Line# 1 7

104 C

105 C

106 SUBROUTINE SOLVE(X,V,A,N,H,W)

107 C

108 C

109 C

110 C

111 C

112 C ** This subroutine produces the numerical solution
113 C to the system of equations of motion

114 C

115 C

116 C

117 C

118 C

119 C

120

LOGICAL FLAG

121 REAL*8 X(3), V(3), A(3,10), AA(3,13), U(6,13)

122 REAL*8 WX2, WXW, WXWX3, HD12, F, W

123 COMMON /BLOCK/ AA, U, WX2, WXW, WXWX3, HD12

124 DATA FLAG /.TRUE./

125 C

126 C

127 C pack user supplied nonhomogeneous part of DE into
128 C the higher part of AA

129 C

130 DO 10 I = 1,10

131 DO 10 K = 1,3

132 AA(K,I+3) = A(K,I)

133 10 CONTINUE

134 C

135 C if this is the first call to solve (FLAG = T), then
136 C initialize

137 C

138 IF (FLAG) THEN

139 CALL INNIT(X,V,W,H)

140 FLAG = .FALSE.

141 END IF

142 C

143 C use the Adam-Brashford 3-step method to advance
144 C the solution h time units. Place the solution
145 C back into X and V.

146 C

147 DO 100 I = 4,N+3

148 DO 100 J = 1,6

149 U(J,I) = U(J,I-1) +
150 HD12*(23*F(J,I-1)-16*F(J,I-2)+5*F(J,I-3))

151 100 CONTINUE

152 X(1) = U(1,N+3)

```

Line# 1      7
153          V(1) = U(2,N+3)
154          X(2) = U(3,N+3)
155          V(2) = U(4,N+3)
156          X(3) = U(5,N+3)
157          V(3) = U(6,N+3)
158 C
159 C          reset U and AA for the next call to SOLVE
160 C
161          DO 200 J = 1,6
162              DO 200 I = 1,3
163                  U(J,I) = U(J,N+I)
164                  IF (J .LE. 3) AA(I,J) = AA(I,N+J)
165          CONTINUE
166 C          DO 300 I = 1,3
167 C              DO 300 K = 1,3
168 C                  AA(K,I) = AA(K,N+I)
169 C300          CONTINUE
170          RETURN
171          END

```

ame	Type	Offset	P	Class
	REAL*8	8	*	
A	REAL*8	0		/BLOCK /
	REAL*8			FUNCTION
AG	LOGICAL*4	496		
	REAL	16	*	
D12	REAL*8	960		/BLOCK /
	INTEGER*4	500		
	INTEGER*4	512		
	INTEGER*4	504		
	INTEGER*4	12	*	
	REAL*8	312		/BLOCK /
	REAL*8	4	*	
	REAL*8	20	*	
2	REAL*8	936		/BLOCK /
XW	REAL*8	944		/BLOCK /
XWX3	REAL*8	952		/BLOCK /
	REAL*8	0	*	

D Line# 1 7

173 C

174 C

175 SUBROUTINE INNIT(X,V,W,H)

176 C

177 C

178 C-----

179 C

180 C

181 C This is the initialization routine which is called only once

182 C

183 C

184 C-----

185 C

186 C

187 REAL * 8 X(3), V(3), AA(3,13), U(6,13), WX2, WXW, WXWX3

188 REAL * 8 CWT, SWT, T, W, HD12

189 COMMON /BLOCK/ AA, U, WX2, WXW, WXWX3, HD12

190 WXW = W*W

191 WXWX3 = 3*WXW

192 WX2 = 2*W

193 HD12 = DBLE(H)/12.0

194 C

195 DO 100 I = 1,3

196 DO 100 J = 1,6

197 AA(J,I) = 0.0

198 100 CONTINUE

199 DO 200 K = 1,3

200 U(2*K-1,3) = X(K)

201 U(2*K,3) = V(K)

202 200 CONTINUE

203 C

204 DO 300 I = 1,2

205 T = H*(I-3)

206 CWT = DCOS(W*T)

207 SWT = DSIN(W*T)

208 U(1,I) = X(1) + V(1)*(4*SWT-3*W*T)/W +

209 + 6*X(3)*(SWT-W*T) + 2*V(3)*(CWT-1.0)/W

210 U(2,I) = V(1)*(4*CWT-3.0) + 6*W*X(3)*(CWT-1.0) -

211 + 2*V(3)*SWT

212 U(3,I) = X(2)*CWT + V(2)*SWT/W

213 U(4,I) = -X(2)*W*SWT + V(2)*CWT

214 U(5,I) = 2*V(1)*(1.0-CWT)/W + X(3)*(4.0-3*CWT) +

215 + V(3)*SWT/W

216 U(6,I) = 2*V(1)*SWT + 3*X(3)*W*SWT + V(3)*CWT

217 300 CONTINUE

218 RETURN

219 END

<< A D A M >>

Page 7

07-05-84

21:33:40

Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

e	Type	Offset	P	Class
	REAL*8	0		/BLOCK /
	REAL*8	560		
				INTRINSIC
				INTRINSIC
				INTRINSIC
	REAL	12	*	
	REAL*8	960		/BLOCK /
	INTEGER*4	540		
	INTEGER*4	544		
	INTEGER*4	548		
	REAL*8	568		
	REAL*8	552		
	REAL*8	312		/BLOCK /
	REAL*8	4	*	
	REAL*8	8	*	
	REAL*8	936		/BLOCK /
	REAL*8	944		/BLOCK /
	REAL*8	952		/BLOCK /
	REAL*8	0	*	

220 \$PAGE

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

221 C

222 C

223 FUNCTION F(J,I)

224 C

225 C

226 C-----

227 C

228 C

229 C User supplied function

230 C

231 C

232 C-----

233 C

234 C

235 REAL*8 AA(3,13),U(6,13),WX2,WXW,WXWX3,HD12,F

236 COMMON /BLOCK/ AA,U,WX2,WXW,WXWX3,HD12

237 C

238 GO TO (10,20,30,40,50,60), J

239 10

CONTINUE

240 F = U(2,I)

241 RETURN

242 20

CONTINUE

243 F = -WX2*U(6,I) + AA(1,I)

244 RETURN

245 30

CONTINUE

246 F = U(4,I)

247 RETURN

248 40

CONTINUE

249 F = -WXW*U(3,I) + AA(2,I)

250 RETURN

251 50

CONTINUE

252 F = U(6,I)

253 RETURN

254 60

CONTINUE

255 F = WX2*U(2,I) + WXWX3*U(5,I) + AA(3,I)

256 RETURN

257 END

Name	Type	Offset	P	Class
AA	REAL*8	0		/BLOCK /
HD12	REAL*8	960		/BLOCK /
I	INTEGER*4	4	*	
J	INTEGER*4	0	*	
U	REAL*8	312		/BLOCK /
WX2	REAL*8	936		/BLOCK /
WXW	REAL*8	944		/BLOCK /
WXWX3	REAL*8	952		/BLOCK /
258 \$PAGE				

Line# 1 7

259 C

260 C

261 SUBROUTINE GETINT (X, V, N)

262 C

263 C

264 C

265 C

266 C

267 C get initial conditions

268 C

269 C

270 C

271 C

272 C

273 REAL * 8 X(N), V(N)

274 C

275 OPEN (1, FILE = 'INITCON.DAT')

276 DO 100 K = 1, N

277 READ (1, 10) X(K)

278 100 CONTINUE

279 C

280 DO 200 K = 1, N

281 READ (1,10) V(K)

282 200 CONTINUE

283 RETURN

284 10 FORMAT (F15.6)

285 END

me Type Offset P Class

INTEGER*4 576

INTEGER*4 8 *

REAL*8 4 *

REAL*8 0 *

Name	Type	Size	Class
LOCK		968	COMMON
EXACT			SUBROUTINE
ETINT	REAL*8		FUNCTION
INNIT			SUBROUTINE
MAIN			SUBROUTINE
OLVE			PROGRAM
			SUBROUTINE

<< A D A M >>

Page 10

07-05-84

21:33:40

D Line# 1 7

Microsoft FORTRAN77 V3.13 8/05/83

Pass One No Errors Detected
285 Source Lines

End of Document